Chapter 10
Numerical Implementations of Comprehensive Grid Generators

10.1 One-Dimensional Equation

Here, we consider a curve S^{x1} specified by parametrization from a normalized parametric interval $S^1 = [0, 1]$

$$\mathbf{x}(s) : [0, 1] \to R^n \ , \quad \mathbf{x} = (x^1, \ldots, x^n) \ . \tag{10.1}$$

For generating a grid on the curve S^{x1}, we first define a grid on the parametric interval $[0, 1]$ with the use of the one-dimensional inverted Beltrami equation in a divergent form. Then, the grid nodes on the parametric interval $[0, 1]$ are mapped with the parametric transformations $\boldsymbol{x}(s)$ on S^{x1}, thus forming a grid on the curve S^{x1}.

The grid nodes in $[0, 1]$ are computed by numerically solving the Dirichlet boundary value problem with respect to an intermediate transformation

$$s(\xi) : [0, 1] \to [0, 1]$$

for Eq. (9.133), i.e.

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\sqrt{g^s}\frac{\mathrm{d}s}{\mathrm{d}\xi}\right) = 0 \ , \quad 0 < \xi < 1 \ , \\ &s(0) = 0 \ , \quad s(1) = 1 \ , \end{aligned} \tag{10.2}$$

where g^s is the determinant of a control covariant metric g^s_{11} over the curve S^{x1}, in particular, specified in the form (9.116) for $n = 1$, i.e.

$$\begin{aligned} &g^s_{11} = z(s)g^{xs}_{11} + F^m(s)F^m(s) \ , \quad m = 1, \ldots, l \ , \\ &g^{xs}_{11} = \frac{\partial \mathbf{x}}{\partial s} \cdot \frac{\partial \mathbf{x}}{\partial s} . \end{aligned}$$

It is evident that in the one-dimensional case, $g^s = g^s_{11}$, $g^{11}_s = 1/g^s_{11}$.

V.D. Liseikin, *Grid Generation Methods*, Scientific Computation,
DOI 10.1007/978-3-319-57846-0_10

The metric g^s_{11} can also be the metric of a monitor curve S^{r1} prescribed by a monitor function for controlling grid properties

$$\mathbf{f}(\mathbf{x}) : G^n \to R^l \ , \quad \mathbf{f} = (f^1, \ldots, f^l) \ ,$$

where G^n is a domain in R^n containing S^{x1}. As a result, the monitor curve S^{r1} over S^{x1} is parametrized by

$$\mathbf{r}(s) : [0, 1] \to R^{n+l} \ , \quad \mathbf{r}(s) = (\mathbf{x}(s), \mathbf{f}[\mathbf{x}(s)]) \ ,$$

and consequently

$$g^s_{11} = g^{rs}_{11} = \mathbf{r}_s \cdot \mathbf{r}_s = \mathbf{x}_s \cdot \mathbf{x}_s + \mathbf{f}_s \cdot \mathbf{f}_s = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} + \frac{\mathrm{d}\mathbf{f}[\mathbf{x}(s)]}{\mathrm{d}s} \cdot \frac{\mathrm{d}\mathbf{f}[\mathbf{x}(s)]}{\mathrm{d}s} \ .$$

10.1.1 Numerical Algorithm

The grid nodes $\mathbf{x}_j$, $j = 0, 1, \ldots, N$, on S^{x1} are defined by the relation

$$\mathbf{x}_j = \mathbf{x}(s(jh)) \ , \quad j = 0, 1, \ldots, N \ , \quad h = 1/N \ ,$$

or by

$$\mathbf{x}_j = \mathbf{x}(s_j) \ , \quad j = 0, 1, \ldots, N \ , \quad h = 1/N \ ;$$

here, s_j, $j = 0, 1, \ldots, N$, is a difference function obtained by the numerical solution on a uniform grid $\xi_j = jh$, $j = 0, 1, \ldots, N$, of the Dirichlet problem (10.2).

Iterative Scheme

The nonlinear problem (10.2) is solved through an iterative process which is engendered by the numerical solution of the following parabolic problem with respect to a function $s(\xi, t)$:

$$\begin{aligned} &\frac{\partial s}{\partial t} - \frac{\partial}{\partial \xi}\left(\sqrt{g^s}\frac{\partial s}{\partial \xi}\right) = 0 \ , \quad 0 \le \xi \le 1 \ , \quad 0 \le t \le T \ , \\ &s(0, t) = 0 \ , \quad s(1, t) = 1 \ , \quad s(\xi, 0) = s_0(\xi) \ . \end{aligned} \tag{10.3}$$

The problem (10.3) is approximated on the uniform grid $(ih, k\tau)$ with respect to s^k_i, $i = 0, 1, \ldots, N$, $k = 0, 1, \ldots$, by the following natural stencil:

$$\begin{aligned} &\frac{s^{k+1}_i - s^k_i}{\tau} = \frac{1}{h^2}[v^k_{i+1/2}(s^{k+1}_{i+1} - s^{k+1}_i) - v^k_{i-1/2}(s^{k+1}_i - s^{k+1}_{i-1})] \ , \\ &s^k_0 = 0 \ , \quad s^k_N = 1 \ , \quad s^0_i = s_0(ih) \ , \quad h = 1/N \ , \end{aligned} \tag{10.4}$$

where

$$v_{i+1/2}^k = \frac{1}{2}\left(\sqrt{g^s(s_i^k)} + \sqrt{g^s(s_{i+1}^k)}\right), \quad i = 0, 1, \ldots, N-1. \tag{10.5}$$

The scheme (10.4) is implicit. Its solution is obtained from the algorithm which is elucidated by the application to the following well-known difference reference problem:

$$\begin{aligned} &A_i^{k+1} s_{i-1}^{k+1} - C_i^{k+1} s_i^{k+1} + B_i^{k+1} s_{i+1}^{k+1} = -F_i^k, \quad i = 1, 2, \ldots, N-1, \\ &s_0^{k+1} = a, \quad s_N^{k+1} = b. \end{aligned} \tag{10.6}$$

The solution to (10.6) is found through the following recursive formulas:

$$s_i^{k+1} = \alpha_{i+1}^{k+1} s_{i+1}^{k+1} + \beta_{i+1}^{k+1}, \quad i = 1, \ldots, N-1, \quad s_N^{k+1} = b, \tag{10.7}$$

where

$$\begin{aligned} \alpha_{i+1}^{k+1} &= \frac{B_i^{k+1}}{C_i^{k+1} - \alpha_i^{k+1} A_i^{k+1}}, \quad i = 1, \ldots, N-1, \quad \alpha_1^{k+1} = 0, \\ \beta_{i+1}^{k+1} &= \frac{A_i^{k+1} \beta_i^{k+1} + F_i^k}{C_i^{k+1} - \alpha_i^{k+1} A_i^{k+1}}, \quad i = 1, \ldots, N-1, \quad \beta_1^{k+1} = a. \end{aligned} \tag{10.8}$$

Thus, assuming in (10.6) $a = 0$, $b = 1$, and

$$\begin{aligned} &A_i^{k+1} = v_{i-1/2}^k, \quad B_i^{k+1} = v_{i+1/2}^k, \quad C_i^{k+1} = v_{i-1/2}^k + v_{i+1/2}^k + \theta, \\ &F_i^k = \theta s_i^k, \quad \theta = h^2/\tau, \quad i = 1, \ldots, N-1, \end{aligned} \tag{10.9}$$

we obtain a solution of (10.4) at a step $k+1$ if it is known at the previous step k. Note that the values of the initial function s_i^0, $i = 0, 1, \ldots, N$, are specified by the user. Naturally, it may be assumed that

$$s_i^0 = ih, \quad i = 0, \ldots, N, \quad h = 1/N.$$

As an approximate numerical solution of (10.3), the solution s_i^k, $i = 0, 1, \ldots, N$, of (10.4) at a step number k is taken if

$$\max_{0 \le i \le N} \frac{|s_i^{k+1} - s_i^k|}{\tau} \le \varepsilon, \tag{10.10}$$

for some sufficiently small $\varepsilon > 0$.

Step–by–Step Algorithm

The algorithm described above is presented here in a step-by-step manner.

Step 1.
Define an initial grid distribution of the parametric interval [0,1] by introducing a monotone difference function s_i^0, $i = 0, \ldots, N$, such that $s_0^0 = 0$, $s_N^0 = 1$.
Step 2.
Compute the difference function $v_{i+1/2}^0$, $i = 0, \ldots, N-1$, by formula (10.5).
Step 3.
Compute the difference functions A_i^1, B_i^1, C_i^1, F_i^0, $i = 1, \ldots, N-1$, by formulas in (10.9).
Step 4.
Compute the coefficients α_i^1 and β_i^1, $i = 1, \ldots, N$, by formulas in (10.8) with $a = 0$.
Step 5.
Compute the difference solution s_i^1, $i = 0, \ldots, N$, of the first step through the formula (10.7), taking into account $s_0^1 = 0$, $s_N^1 = b = 1$.
Step 6.
Return to step 2 assuming $s_0^0 = s_i^1$, $i = 0, \ldots, N$, where s_i^1 is the solution obtained at step 5.

Continue until the tolerance requirement (10.10) is observed.
Step 7.
Map the final nodes s_i^k, $i = 0, \ldots, N$, satisfying (10.10), with the parametrization $\boldsymbol{x}(s)$ on S^{x1}.

The algorithm described is readily reformulated for the numerical solution of the inverted diffusion equation in a divergent form, namely, by substituting $w(\mathbf{s})$ for $\sqrt{g^{\mathbf{s}}}$ in (10.2), (10.3), and (10.5).

10.2 Multidimensional Finite Difference Algorithms

In this section, we apply one version of the multidimensional algorithm of fractional steps proposed by Yanenko (1971) for the numerical solution of the inverted n-dimensional ($n \geq 2$) Beltrami and diffusion equations. Other versions of this algorithm that can be readily implemented for solving the resulting multidimensional grid equations, in particular, the popular ADI (alternating direction implicit) method are reviewed by Kovenya et al. (1990), Fletcher (1997), and Langtangen (2003).

10.2.1 Parabolic Simulation

We rewrite the boundary value problems (9.132) and (9.134) in the following general form:

$$\begin{aligned} &B_n^{\boldsymbol{\xi}}[s^i] = R^i[\boldsymbol{s}], \quad i = 1, \ldots, n, \\ &s^i(\boldsymbol{\xi}) = \psi^i(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \partial \Xi^n, \end{aligned} \tag{10.11}$$

where

$$B_n^{\boldsymbol{\xi}}[s^l] = g^{\boldsymbol{\xi}} g_{\boldsymbol{\xi}}^{ij} \frac{\partial^2 s^l}{\partial \xi^i \partial \xi^j} , \quad i, j, l = 1, \ldots n ,$$
$$g^{\boldsymbol{\xi}} = \det\{g_{ij}^{\boldsymbol{\xi}}\} = (J)^2 g^{\boldsymbol{s}} = 1/\det\{g_{\boldsymbol{\xi}}^{ij}\} .$$

For inverted Beltrami equations (9.132) in a general control metric $g_{ij}^{\mathbf{s}}$, we have in (10.11)

$$R^i[\boldsymbol{s}] = (J)^2 \sqrt{g^{\mathbf{s}}} \frac{\partial}{\partial \xi^j} (\sqrt{g^{\mathbf{s}}} g_{\mathbf{s}}^{im}) \frac{\partial \xi^j}{\partial s^m} , \quad i, j, m = 1, \ldots, n . \tag{10.12}$$

Notice that for the metric (9.9) of a monitor surface over a domain S^n, i.e.

$$g_{ij}^{\mathbf{s}} = g_{ij}^{rs} = \delta_j^i + \frac{\partial \boldsymbol{f}(\boldsymbol{s})}{\partial s^i} \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{s})}{\partial s^j} , \quad i, j = 1, \ldots, n ,$$

in accordance with (9.56), formula (10.12) also has the following form:

$$R^i[\boldsymbol{s}] = -B_n^{r\boldsymbol{\xi}}[\boldsymbol{f}] \cdot \frac{\partial \boldsymbol{f}[\boldsymbol{s}(\boldsymbol{\xi})]}{\partial \xi^j} \frac{\partial \xi^j}{\partial s^i} , \quad i, j = 1, \ldots, n , \tag{10.13}$$

where

$$B_n^{\boldsymbol{\xi}}[y] = B_n^{r\xi}[y] = g^{r\xi} g_{\xi r}^{ij} \frac{\partial^2 y}{\partial \xi^i \partial \xi^j} , \quad i, j, l = 1, \ldots n ,$$
$$g^{r\xi} = \det\{g_{ij}^{r\xi}\} = (J)^2 g^{rs} = 1/\det\{g_{\xi r}^{ij}\} ,$$
$$g_{ij}^{r\xi} = g_{kl}^{rs} \frac{\partial s^k}{\partial \xi^i} \frac{\partial s^l}{\partial \xi^j} = g_{ij}^{s\xi} + \frac{\partial \boldsymbol{f}[\boldsymbol{s}(\boldsymbol{\xi})]}{\partial \xi^i} \cdot \frac{\partial \boldsymbol{f}[\boldsymbol{s}(\boldsymbol{\xi})]}{\partial \xi^j} , \quad i, j, k, l = 1, \ldots, n .$$

For the general inverted diffusion equations (9.134), we have in (10.11)

$$R^i[\boldsymbol{s}] = \frac{g^{\mathbf{s}}(J)^2}{w(s)} \frac{\partial}{\partial \xi^k} (w(\boldsymbol{s}) g_{\mathbf{s}}^{ij}) \frac{\partial \xi^k}{\partial s^j} , \quad i, j, k = 1, \ldots, n , \tag{10.14}$$

in particular, for (13.49), i.e. when $w(\boldsymbol{s}) = 1$, $g_s^{ij} = Z[v](\boldsymbol{s}) \delta_j^i$,

$$R^i = \frac{J^2}{Z[v](\boldsymbol{s})} \frac{\partial}{\partial s^i} Z[v](\boldsymbol{s}) , \qquad i = 1, \ldots, n , \tag{10.15}$$

$$B_n^{\boldsymbol{\xi}}[y] = B_n^{s\xi}[y] = g^{s\xi} g_{\xi s}^{ij} \frac{\partial^2 y}{\partial \xi^i \partial \xi^j} , \quad g^{s\xi} = \det\{g_{ij}^{s\xi}\} = J^2 , \quad i, j = 1, \ldots, n ,$$
$$g_{ij}^{s\xi} = \frac{\partial \boldsymbol{s}}{\partial \xi^i} \cdot \frac{\partial \boldsymbol{s}}{\partial \xi^j} = \frac{\partial s^k}{\partial \xi^i} \frac{\partial s^k}{\partial \xi^j} , \qquad g_{\xi s}^{ij} = \frac{\partial \xi^i}{\partial s^k} \frac{\partial \xi^j}{\partial s^k} , \qquad i, j, k = 1, \ldots, n . \tag{10.16}$$

Solutions to the non-linear boundary value problem (10.11) may be found in the following way. First, the problem is replaced by a nonstationary boundary value problem with respect to the components $s^i(\boldsymbol{\xi}, t)$, $i = 1, \ldots, n$, of the vector function $\boldsymbol{s}(\boldsymbol{\xi}, t) : \Xi^n \times [0, T] \to S^n$:

$$\begin{aligned} &\frac{\partial s^i}{\partial t} = (J)^p \left\{ B_n^{\boldsymbol{\xi}}[s^i] - R^i[\boldsymbol{s}] \right\}, \quad i, j, m = 1, \ldots, n \,, \\ &s^i(\boldsymbol{\xi}, t) = \psi^i(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \partial\Xi^n, \quad t \geq 0 \,, \\ &s^i(\boldsymbol{\xi}, 0) = s_0^i(\boldsymbol{\xi}) \,, \quad \boldsymbol{\xi} \in \Xi^n \,, \end{aligned} \tag{10.17}$$

where $J = \det\{\partial s^i / \partial \xi^j\}$, $p \geq 0$, $s_0^i(\boldsymbol{\xi})$ is the i-th component of the initial transformation

$$\mathbf{s}_0(\boldsymbol{\xi}) : \Xi^n \to S^n, \quad \mathbf{s}_0(\boldsymbol{\xi}) = [s_0^1(\boldsymbol{\xi}), \ldots, s_0^n(\boldsymbol{\xi})]$$

specified by the user. Then, for an approximate solution $\boldsymbol{s}(\boldsymbol{\xi})$ of (10.11), there can be taken the solution $\boldsymbol{s}(\boldsymbol{\xi}, t)$ of (10.17) for some sufficiently large t.

If the elements $g_{ij}^{\mathbf{s}}$ are known at all points of S^n, then in (10.17), $p = 0$, and for (10.12), (10.13), (10.14), and (10.15), we can assume, respectively,

$$R^i[\boldsymbol{s}] = (J)^2 \sqrt{g^{\mathbf{s}}} \frac{\partial}{\partial s^m} (\sqrt{g^{\mathbf{s}}} g_{\mathbf{s}}^{im}) \,, \quad i, m = 1, \ldots, n \,, \tag{10.18}$$

$$R^i[\boldsymbol{s}] = -B_n^{r\boldsymbol{\xi}}[\boldsymbol{f}(\boldsymbol{s})] \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{s})}{\partial s^i} \,, \quad i = 1, \ldots, n \,, \tag{10.19}$$

$$R^i[\boldsymbol{s}] = \frac{g^{\mathbf{s}}(J)^2}{w(s)} \frac{\partial}{\partial s^j} (w(\boldsymbol{s}) g_{\mathbf{s}}^{ij}) \,, \quad i, j = 1, \ldots, n \,, \tag{10.20}$$

$$R^i = \frac{J^2}{Z[v](\boldsymbol{s})} \frac{\partial}{\partial s^i} Z[v](\boldsymbol{s}) \,, \quad i = 1, \ldots, n \,. \tag{10.21}$$

When $B_n^{\boldsymbol{\xi}}$ is an elliptic operator, the solution to the problem (10.17) relaxes to the solution of (10.11) as $t \to \infty$.

The factor $(J)^p$, $p \geq 1$ in (10.17) is introduced in the case when the control metric $g_{ij}^{\mathbf{s}}$ is not known in advance, but is found numerically, for instance, if it is dependent on the solution of the physical problem for which the numerical grid is generated. This factor allows one to rule out the Jacobian J being a denominator after replacing in $R^i[\boldsymbol{s}]$ the derivatives $\partial \xi^i / \partial s^j$ with the derivatives $\partial s^k / \partial \xi^m$. In particular, in the case of the metric of a monitor hypersurface S^{rn} over S^n (see (9.13) and (9.14)), i.e. when $g_{ij}^{\mathbf{s}} = g_{ij}^{r\xi}$, it is sufficient to assume $p = 1$. Namely, for $n = 2$, we have, from (10.13) and (2.4),

$$
\begin{aligned}
J R^i[\boldsymbol{s}] &= -J B_2^{r\xi}[\boldsymbol{f}(\boldsymbol{s})] \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{s}(\boldsymbol{\xi}))}{\partial \xi^m} \frac{\partial \xi^m}{\partial s^i} \\
&= -(-1)^{i+m} B_2^{r\xi}[\boldsymbol{f}(\boldsymbol{s})] \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{s}(\boldsymbol{\xi}))}{\partial \xi^m} \frac{\partial s^{3-i}}{\partial \xi^{3-m}} , \quad i, m = 1, 2 ,
\end{aligned}
\tag{10.22}
$$

while for $n = 3$, we obtain, from (10.13) and (2.5),

$$
\begin{aligned}
J R^i[\boldsymbol{s}] &= -J B_3^{r\xi}[\boldsymbol{f}(\boldsymbol{s})] \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{s}(\boldsymbol{\xi}))}{\partial \xi^m} \frac{\partial \xi^m}{\partial s^i} \\
&= -B_3^{r\xi}[\boldsymbol{f}(\boldsymbol{s})] \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{s}(\boldsymbol{\xi}))}{\partial \xi^m} \left(\frac{\partial s^{i+1}}{\partial \xi^{m+1}} \frac{\partial s^{i+2}}{\partial \xi^{m+2}} - \frac{\partial s^{i+1}}{\partial \xi^{m+2}} \frac{\partial s^{i+2}}{\partial \xi^{m+1}} \right) , \\
& \quad i, m = 1, 2, 3 .
\end{aligned}
\tag{10.23}
$$

With such incorporation of $(J)^p$, one can produce a final nondegenerate grid even if the initial and intermediate grids may be singular. Note that the numerical implementations of the inverted energy and diffusion functionals cannot eliminate the Jacobian being the denominator.

The boundary value problem (10.17) is usually solved through alternative direction implicit methods, in particular, through the method of fractional steps.

10.2.2 Two-Dimensional Equations

In this section, a finite-difference numerical algorithm for generating grids in two-dimensional domains and surfaces is described.

Boundary Value Problem

Let us first discuss the grid algorithm for a two-dimensional domain S^2. We shall use, for the logical domain Ξ^2, the unit square: $\Xi^2 = \{0 \le \xi^1, \xi^2 \le 1\}$. Let the transformation $\mathbf{s}(\boldsymbol{\xi})$ for generating a grid in S^2 be specified on the boundary of Ξ^2, i.e. there is a map

$$
\boldsymbol{\varphi}(\boldsymbol{\xi}) : \partial \Xi^2 \to \partial S^2 , \quad \boldsymbol{\varphi} = (\varphi^1, \varphi^2) \tag{10.24}
$$

which is continuous on $\partial \Xi^2$. Note that the one-dimensional transformation on any segment of $\partial \Xi^2$ can be computed by the algorithm described in Sect. 11.1. We consider here the generation of a grid in S^2 by the numerical solution of the Dirichlet problem (10.11) for the most general system of inverted Beltrami equations in a control metric g^s_{ij} for $n = 2$ written in a vector form

$$
\begin{aligned}
& B_2^{\xi}[\boldsymbol{s}] = \boldsymbol{R}[\boldsymbol{s}] , \\
& \boldsymbol{s}(\boldsymbol{\xi}) \Big|_{\partial \Xi^2} = \boldsymbol{\varphi}(\boldsymbol{\xi}), \quad i = 1, 2 ,
\end{aligned}
\tag{10.25}
$$

where

$$\begin{aligned}
&\boldsymbol{s}(\boldsymbol{\xi}) = (s^1(\boldsymbol{\xi})\ ,\ s^2(\boldsymbol{\xi}))\ ,\quad \boldsymbol{\varphi}(\boldsymbol{\xi}) = (\varphi^1(\boldsymbol{\xi}),\ \varphi^2(\boldsymbol{\xi}))\ ,\quad \boldsymbol{R}[\boldsymbol{s}] = (R^1[\boldsymbol{s}]\ ,\ R^2[\boldsymbol{s}])\ ,\\
&B_2^{\boldsymbol{\xi}}[\boldsymbol{s}] = g_{22}^{\boldsymbol{\xi}}\frac{\partial^2 \boldsymbol{s}}{\partial\xi^1\partial\xi^1} - 2g_{12}^{\boldsymbol{\xi}}\frac{\partial^2 \boldsymbol{s}}{\partial\xi^1\partial\xi^2} + g_{11}^{\boldsymbol{\xi}}\frac{\partial^2 \boldsymbol{s}}{\partial\xi^2\partial\xi^2}\ ,\\
&R^1[\boldsymbol{s}] = J\sqrt{g^{\mathbf{s}}}\Big[\frac{\partial}{\partial\xi^1}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{11})\frac{\partial s^2}{\partial\xi^2} - \frac{\partial}{\partial\xi^1}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{12})\frac{\partial s^1}{\partial\xi^2}\\
&\quad + \frac{\partial}{\partial\xi^2}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{12})\frac{\partial s^1}{\partial\xi^1} - \frac{\partial}{\partial\xi^2}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{11})\frac{\partial s^2}{\partial\xi^1}\Big]\ ,\\
&R^2[\boldsymbol{s}] = J\sqrt{g^{\mathbf{s}}}\Big[\frac{\partial}{\partial\xi^1}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{11})\frac{\partial s^2}{\partial\xi^2} - \frac{\partial}{\partial\xi^1}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{12})\frac{\partial s^1}{\partial\xi^2}\\
&\quad + \frac{\partial}{\partial\xi^2}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{12})\frac{\partial s^1}{\partial\xi^1} - \frac{\partial}{\partial\xi^2}(\sqrt{g^{\mathbf{s}}}g_{\mathbf{s}}^{11})\frac{\partial s^2}{\partial\xi^1}\Big]\ .
\end{aligned} \tag{10.26}$$

Parabolic Equations

The nonlinear boundary value problem (10.25) is solved by an iterative process. For this purpose, in accordance with (10.17), the problem (10.25) is replaced by the following boundary value parabolic problem with respect to the function $\boldsymbol{s}(\xi^1, \xi^2, t) = (s^1(\xi^1, \xi^2, t),\ s^2(\xi^1, \xi^2, t))$:

$$\begin{aligned}
&\frac{\partial \boldsymbol{s}}{\partial t} = (J)^p\Big\{B_2^{\boldsymbol{\xi}}[\boldsymbol{s}] - \boldsymbol{R}(\boldsymbol{s})\Big\}\ ,\\
&\boldsymbol{s}(\boldsymbol{\xi}, t) = \boldsymbol{\varphi}(\boldsymbol{\xi})\ ,\quad \boldsymbol{\xi} \in \partial\Xi^2\ ,\quad t \geq 0\ ,\\
&\boldsymbol{s}(\boldsymbol{\xi}, 0) = \boldsymbol{s}_0(\boldsymbol{\xi})\ ,\quad \boldsymbol{\xi} \in \Xi^2\ ,
\end{aligned} \tag{10.27}$$

where $\boldsymbol{s}_0(\boldsymbol{\xi})$ is an initial transformation

$$\mathbf{s}_0(\boldsymbol{\xi}) : \Xi^2 \to S^2,\quad \mathbf{s}_0(\boldsymbol{\xi}) = [s_0^1(\boldsymbol{\xi}), s_0^2(\boldsymbol{\xi})],$$

specified by the user.

The solution $\mathbf{s}(\boldsymbol{\xi}, t)$ satisfying (10.27) aspires to the solution to (10.25) when $t \to \infty$. Therefore, an approximate solution of (10.25) is obtained from the solution to (10.27) computed for some sufficiently large value $t = T_0$.

Initial Transformation

The initial transformation for (10.27)

$$\mathbf{s}(\boldsymbol{\xi}, 0) = \mathbf{s}_0(\boldsymbol{\xi}) : \Xi^2 \to S^2.$$

can be found by propagating the values of $\boldsymbol{\varphi}(\boldsymbol{\xi}) = [\varphi^1(\boldsymbol{\xi}), \varphi^2(\boldsymbol{\xi})]$ from the boundary points into the interior of the domain Ξ^2, for example, if Ξ^2 is the unit square through the formula of the Lagrange two-dimensional transfinite interpolation described in Chap. 5. This formula has the following recursive form for the components $s^i(\boldsymbol{\xi}, 0)$ of the mapping $\mathbf{s}_0(\boldsymbol{\xi})$:

$$\begin{aligned} F^i(\xi^1,\xi^2) &= \alpha^i_{01}(\xi^1)\varphi^i(0,\xi^2)+\alpha^i_{11}(\xi^1)\varphi^i(1,\xi^2)\,,\\ s^i(\xi^1,\xi^2,0) &= F^i(\xi^1,\xi^2)+\alpha^i_{02}(\xi^2)[\varphi^i(\xi^1,0)-F^i(\xi^1,0)]\\ &\quad +\alpha^i_{12}(\xi^2)[\varphi^i(\xi^1,1)-F^i(\xi^1,1)]\,,\quad i=1,2,\quad i \text{ fixed}\,, \end{aligned} \tag{10.28}$$

where the functions $\alpha^i_{kj}(s)$, $0 \le s \le 1$, (referred to as blending functions) are subject to the following restrictions:

$$\alpha^i_{0j}(0)=\alpha^i_{1j}(1)=1\,,\quad \alpha^i_{0j}(1)=\alpha^i_{1j}(0)=0\,. \tag{10.29}$$

In particular, for the simplest expressions of the blending functions

$$\alpha^i_{0j}(s)=1-s\,,\quad \alpha^i_{1j}(s)=s\,,$$

satisfying (10.29), we find, from (10.28),

$$\begin{aligned} F^i(\xi^1,\xi^2) &= (1-\xi^1)\varphi^i(0,\xi^2)+\xi^1\varphi^i(1,\xi^2)\,,\\ s^i(\xi^1,\xi^2,0) &= F^i(\xi^1,\xi^2)+(1-\xi^2)[\varphi^i(\xi^1,0)-F^i(\xi^1,0)]\\ &\quad +\xi^2[\varphi^i(\xi^1,1)-F^i(\xi^1,1)]\,,\quad i=1,2\,. \end{aligned} \tag{10.30}$$

Iterative Algorithm for Generating Quadrilateral Grids

The problem (10.27) is approximated on the rectangular grid $(ih_1, jh_2, k\tau)$, $h_1 = 1/N_1$, $h_2 = 1/N_2$, in the logical domain $\Xi^2 \times [0,T]$, where Ξ^2 is a rectangle (Fig. 10.1 (*left-hand*)), by the scheme

$$\begin{aligned} \frac{\boldsymbol{s}^{k+1/2}-\boldsymbol{s}^k}{\tau/2} &= J^p(\boldsymbol{s}^k)\Big\{g^{\xi}_{22}[\boldsymbol{s}^k]L^h_{11}[\boldsymbol{s}^{k+1/2}]+g^{\xi}_{11}[\boldsymbol{s}^k]L^h_{22}[\boldsymbol{s}^k]\\ &\quad -2g^{\xi}_{12}[\boldsymbol{s}^k]L^h_{12}[\boldsymbol{s}^k]\Big\}-J^p(\boldsymbol{s}^k)\boldsymbol{R}[\boldsymbol{s}^k],\\ \frac{\boldsymbol{s}^{k+1}-\boldsymbol{s}^{k+1/2}}{\tau/2} &= J^p(\boldsymbol{s}^k)\Big\{g^{\xi}_{11}[\boldsymbol{s}^k]L^h_{22}[\boldsymbol{s}^{k+1}]-g^{\xi}_{11}[\boldsymbol{s}^k]L^h_{22}[\boldsymbol{s}^k]\Big\}\,, \end{aligned} \tag{10.31}$$

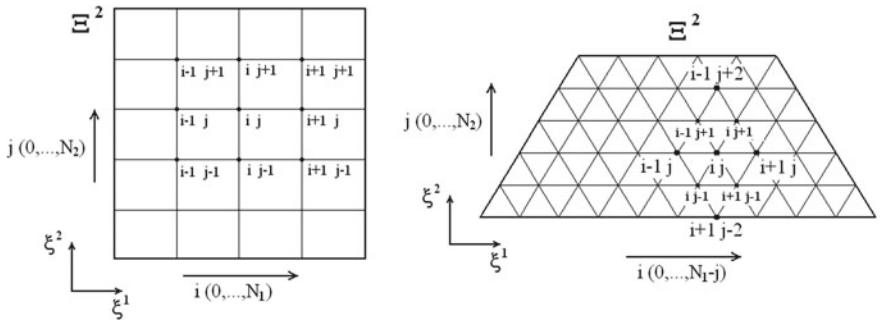

Fig. 10.1 Two-dimensional quadrilateral and triangular stencils for finite differences

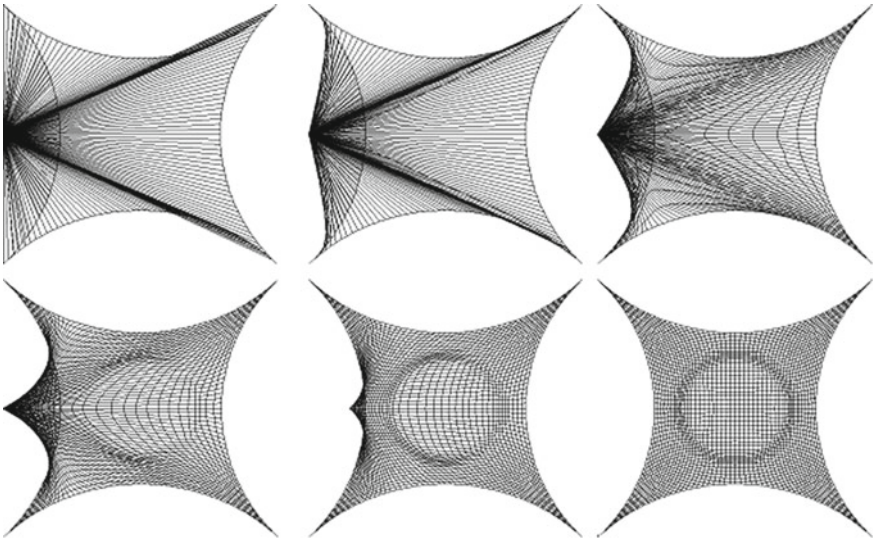

Fig. 10.2 Stages of the iterative generation of a quadrilateral grid with the use of a singular initial grid

where $\boldsymbol{s}^{k+\alpha} = \boldsymbol{s}(\boldsymbol{\xi}, (k + \alpha\tau))$, $k = 0, 1, 2, \ldots$, $\alpha = 0,\ 1/2,\ 1$, L_{ij}^h is a finite-difference operator approximating the operator $\partial^2/(\partial\xi^i \partial\xi^j)$, by the central differences. The derivatives in J, $g_{ij}^{\boldsymbol{\xi}}$, and $\boldsymbol{R}$ are also approximated by the central differences. The initial transformation $\boldsymbol{s}(\boldsymbol{\xi}, 0) = \boldsymbol{s}_0(\boldsymbol{\xi})$ is found through the formulas of transfinite interpolations.

The solution to (10.31) at each step k and $k + 1/2$ is obtained in the same way as it was described in Sect. 11.1.

An approximate solution to (10.27) is the solution $\mathbf{s}_{ij}^k$ at a step k such that

$$\max_{0\le i\le N_1, 0\le j\le N_2} \frac{1}{\tau}|\boldsymbol{s}_{ij}^{k+1} - \boldsymbol{s}_{ij}^k| \le \varepsilon\ , \tag{10.32}$$

for some sufficiently small $\varepsilon > 0$.

Figure 10.2 demonstrates some steps of the grid generation in a two-dimensional domain by the solution of the inverted Beltrami equations with the iterative algorithm described. The initial grid is singular (all its interior nodes merge into one node lying outside of the domain).

Generation of Triangular Grids

The numerical algorithm described above for generating quadrilateral grids is naturally applied to the generation of triangular grids when the logical domain is a symmetric trapezoid (Fig. 10.1 (*right-hand*)).

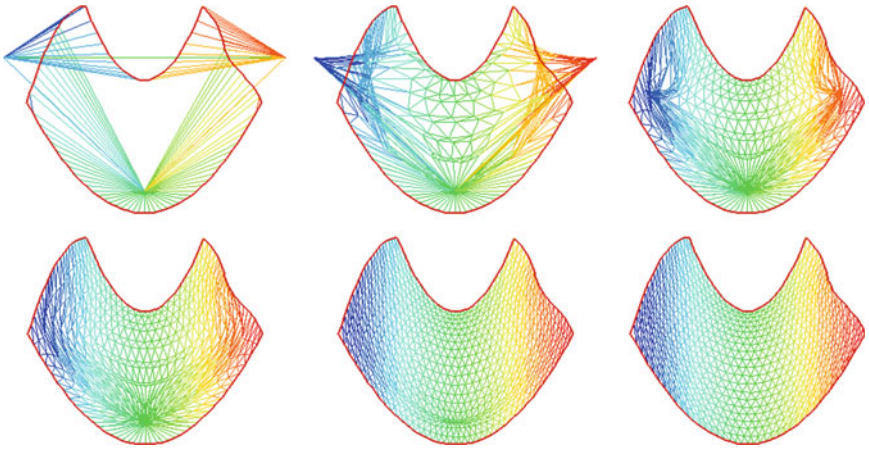

Fig. 10.3 Stages for generating a triangular grid by using a singular initial grid

An example of a triangular grid generated by such an algorithm is exhibited by Fig. 10.3. As it is in Fig. 10.2, the initial grid is singular. All its interior points are placed into three points, two of which lie outside of the domain.

Algorithm for Generating Grids on Two-Dimensional Surfaces

In the same way as for domains, grids are generated in a two-dimensional surface S^{x2} represented as

$$\mathbf{x}(\mathbf{s}) : S^2 \to R^3 , \quad \mathbf{x} = (x^1, x^2, x^3) , \quad \mathbf{s} = (s^1, s^2) , \tag{10.33}$$

by solving the boundary value problem for the inverted two-dimensional diffusion equations as well as for the corresponding inverted Beltrami equations with respect to a monitor metric $g_{ij}^{\mathbf{s}}$ over S^{x2}.

Similarly to the case of a two-dimensional domain, we can choose a rectangle or trapezoid for the logical domain Ξ^2. We can also assume that the boundary transformation

$$\boldsymbol{\varphi}(\boldsymbol{\xi}) : \partial\Xi^2 \to \partial S^2 , \quad \boldsymbol{\varphi} = (\varphi^1, \varphi^2) ,$$

which is continuous on $\partial\Xi^2$, has been specified on the boundary grid points of $\partial\Xi^2$, for example, by computing it through the algorithm described in Sect. 10.1.

The grid on S^{x2} is obtained by mapping, with $\mathbf{x}(\mathbf{s})$, the grid nodes computed in S^2 through the numerical solution of the Dirichlet problem with respect to $\mathbf{s}(\boldsymbol{\xi})$ for the inverted grid equations.

Figure 10.4 illustrates a surface triangular adaptive grid generated by the algorithm.

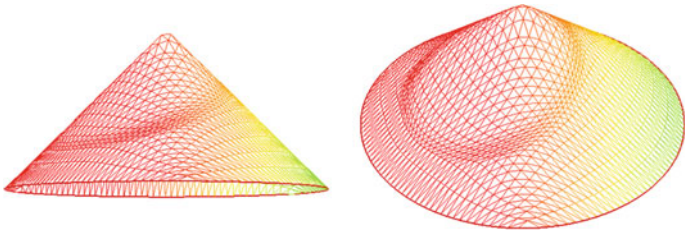

Fig. 10.4 A triangular adaptive grid on a conical surface

10.2.3 Three–Dimensional Problem

For generating grids in a three-dimensional domain $S^3 \subset R^3$ with the use of the inverted Beltrami or diffusion equations in a control metric $g^{\mathbf{s}}_{ij}$, $i, j = 1, 2, 3$, we consider boundary value problem (10.11) for $n = 3$ written in a vector form

$$\begin{aligned} & B_3^{\xi}[\boldsymbol{s}] = \boldsymbol{R}[\boldsymbol{s}] \ , \\ & \boldsymbol{s}(\boldsymbol{\xi}) \Big|_{\partial \Xi^3} = \boldsymbol{\varphi}(\boldsymbol{\xi}) \ , \end{aligned} \tag{10.34}$$

where

$$\begin{aligned} & \boldsymbol{s}(\boldsymbol{\xi}) = (s^1(\boldsymbol{\xi}) \ , \ s^2(\boldsymbol{\xi} \ , \ s^3(\boldsymbol{\xi})) \ , \quad \boldsymbol{\varphi}(\boldsymbol{\xi}) = [\varphi^1(\boldsymbol{\xi}), \varphi^2(\boldsymbol{\xi}), \varphi^3(\boldsymbol{\xi})] \ , \\ & \boldsymbol{R}(\boldsymbol{s}) = (R^1(\boldsymbol{s}) \ , \ R^2(\boldsymbol{s}) \ , \ R^3(\boldsymbol{s})) \ , \end{aligned}$$

$$\begin{aligned} B_3^{\xi}[v] &= g^{\xi} g_{\xi}^{ij} \frac{\partial^2 \boldsymbol{s}}{\partial \xi^i \partial \xi^j} \\ &= [g^{\xi}_{22} g^{\xi}_{33} - (g^{\xi}_{23})^2] \frac{\partial^2 \boldsymbol{s}}{\partial \xi^1 \partial \xi^1} + 2[g^{\xi}_{23} g^{\xi}_{13} - g^{\xi}_{12} g^{\xi}_{33}] \frac{\partial^2 \boldsymbol{s}}{\partial \xi^1 \partial \xi^2} \\ &\quad + 2[g^{\xi}_{12} g^{\xi}_{23} - g^{\xi}_{22} g^{\xi}_{13}] \frac{\partial^2 \boldsymbol{s}}{\partial \xi^1 \partial \xi^3} + [g^{\xi}_{11} g^{\xi}_{33} - (g^{\xi}_{13})^2] \frac{\partial^2 \boldsymbol{s}}{\partial \xi^2 \partial \xi^2} \\ &\quad + 2[g^{\xi}_{12} g^{\xi}_{13} - g^{\xi}_{11} g^{\xi}_{23}] \frac{\partial^2 \boldsymbol{s}}{\partial \xi^2 \partial \xi^3} + [g^{\xi}_{11} g^{\xi}_{22} - (g^{\xi}_{12})^2] \frac{\partial^2 \boldsymbol{s}}{\partial \xi^3 \partial \xi^3} \ , \\ R^i[\boldsymbol{s}] &= J \sqrt{g^{\mathbf{s}}} \frac{\partial}{\partial \xi^j} (\sqrt{g^{\mathbf{s}}} g^{im}_{\mathbf{s}}) \Big(\frac{\partial s^{m+1}}{\partial \xi^{j+1}} \frac{\partial s^{m+2}}{\partial \xi^{j+2}} - \frac{\partial s^{m+1}}{\partial \xi^{j+2}} \frac{\partial s^{m+2}}{\partial \xi^{j+1}} \Big) \ , \\ & i, j, m = 1, 2, 3 \ . \end{aligned} \tag{10.35}$$

Analogously to the solution of two-dimensional problem (10.27), we find a solution to (10.34) as a limit with $t \to \infty$ of the solution of the corresponding parabolic problem

$$\begin{aligned} & \frac{\partial \boldsymbol{s}}{\partial t} = J^p \Big\{ B_3^{\xi}[\boldsymbol{s}] - \boldsymbol{R}(\boldsymbol{s}) \Big\} \ , \\ & \boldsymbol{s}(\boldsymbol{\xi}, t) = \boldsymbol{\varphi}(\boldsymbol{\xi}) \ , \quad \boldsymbol{\xi} \in \partial \Xi^3 \ , \quad t \geq 0 \ , \\ & \boldsymbol{s}(\boldsymbol{\xi}, 0) = \mathbf{s}_0(\boldsymbol{\xi}) \ , \quad \boldsymbol{\xi} \in \Xi^3 \ . \end{aligned} \tag{10.36}$$

Initial Transformation

The initial transformation

$$\boldsymbol{s}(\boldsymbol{\xi}, 0) = \boldsymbol{s}_0(\boldsymbol{\xi}) : \Xi^3 \to S^3 \ .$$

can be found by propagating the values of $\boldsymbol{\varphi}(\boldsymbol{\xi})$ into the interior of the unit cube Ξ^3, for example, through the formula of Lagrange transfinite interpolation. In particular, for the simplest expressions of the blending functions

$$\alpha^i_{0j}(s) = 1 - s \ , \quad \alpha^i_{1j}(s) = s \ ,$$

we find from (5.26)

$$\begin{aligned} &F^i_1(\xi^1, \xi^2, \xi^3) = (1 - \xi^1)\varphi^i(0, \xi^2, \xi^3) + \xi^1\varphi^i(1, \xi^2, \xi^3) \ , \\ &F^i_2(\xi^1, \xi^2, \xi^3) = F^i_1(\xi^1, \xi^2, \xi^3) + (1 - \xi^2)[\varphi^i(\xi^1, 0, \xi^3) \\ &-F^i_1(\xi^1, 0, \xi^3)] + \xi^2[\varphi^i(\xi^1, 1, \xi^3) - F^i_1(\xi^1, 1, \xi^3)] \ , \\ &x^i(\xi^1, \xi^2, \xi^3) = F^i_2(\xi^1, \xi^2, \xi^3) + (1 - \xi^3)[\varphi^i(\xi^1, \xi^2, 0) \\ &-F^i_2(\xi^1, \xi^2, 0)] + \xi^3[\varphi^i(\xi^1, \xi^2, 1) - F^i_2(\xi^1, \xi^2, 1)] \ , \quad i = 1, 2, 3 \ , \end{aligned} \tag{10.37}$$

Three–Dimensional Algorithm

A numerical algorithm for solving problem (10.36) is formulated analogously to the two-dimensional algorithm reviewed by formula (10.31), namely, by splitting the process of the numerical solution on the computational domain Ξ^3, exhibited in Figs. 10.5 and 10.6, into three one-dimensional algorithms:

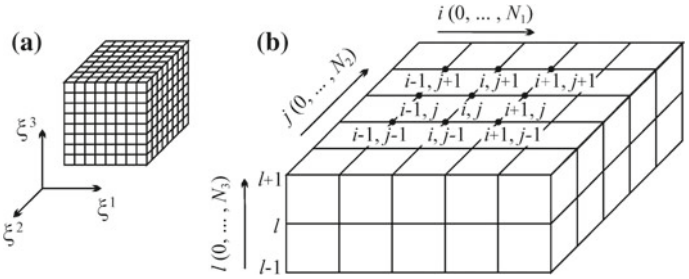

Fig. 10.5 Computational domain Ξ^3 (**a**) and computational stencil (**b**) for generating hexahedral grids

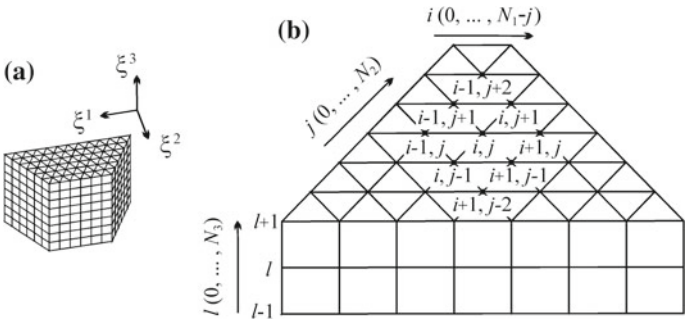

Fig. 10.6 Computational domain Ξ^3 (**a**) and computational stencil (**b**) for generating prismatic grids

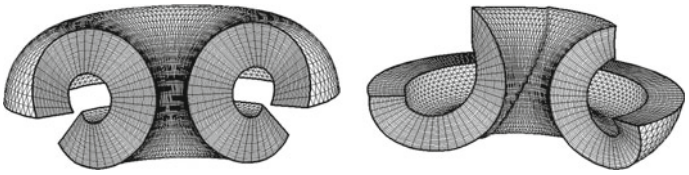

Fig. 10.7 Three-dimensional domain with a prismatic adaptive grid

$$
\begin{aligned}
\frac{\boldsymbol{s}^{k+1/3}-\boldsymbol{s}^{k}}{\tau/3} &= J^{p}(\boldsymbol{s}^{k})\Big\{a^{11}[\boldsymbol{s}^{k}]L^{h}_{11}[\boldsymbol{s}^{k+1/3}]+a^{22}[\boldsymbol{s}^{k}]L^{h}_{22}[\boldsymbol{s}^{k}] \\
&\quad + a^{33}[\boldsymbol{s}^{k}]L^{h}_{33}[\boldsymbol{s}^{k}]+2a^{12}[\boldsymbol{s}^{k}]L^{h}_{12}[\boldsymbol{s}^{k}] \\
&\quad + 2a^{13}[\boldsymbol{s}^{k}]L^{h}_{13}[\boldsymbol{s}^{k}]+2a^{23}[\boldsymbol{s}^{k}]L^{h}_{23}[\boldsymbol{s}^{k}]\Big\} \\
&\quad - J^{p}(\boldsymbol{s}^{k})\boldsymbol{R}[\boldsymbol{s}^{k}]\,, \\
\frac{\boldsymbol{s}^{k+2/3}-\boldsymbol{s}^{k+1/3}}{\tau/3} &= J^{p}(\boldsymbol{s}^{k})\Big\{a^{22}[\boldsymbol{s}^{k}]L^{h}_{22}[\boldsymbol{s}^{k+2/3}]-a^{22}[\boldsymbol{s}^{k}]L^{h}_{22}[\boldsymbol{s}^{k}]\Big\}\,, \\
\frac{\boldsymbol{s}^{k+1}-\boldsymbol{s}^{k+2/3}}{\tau/3} &= J^{p}(\boldsymbol{s}^{k})\Big\{a^{33}[\boldsymbol{s}^{k}]L^{h}_{33}[\boldsymbol{s}^{k+1}]-a^{33}[\boldsymbol{s}^{k}]L^{h}_{33}[\boldsymbol{s}^{k}]\Big\},
\end{aligned}
\tag{10.38}
$$

where $a^{ij} = g^{\xi}g^{ij}_{\xi} = g^{\xi}_{i+1j+1}g^{\xi}_{i+2j+2} - g^{\xi}_{i+1j+2}g^{\xi}_{i+2j+1}$, $i, j = 1, 2, 3$, i, j, k – fixed, $\boldsymbol{s}^{k+\alpha} = \boldsymbol{s}(\boldsymbol{\xi}, (k+\alpha\tau))$, $k = 0, 1, 2, \ldots$, $\alpha = 0, 1/2, 2/3, 1$. L^{h}_{ij} is a finite-difference operator approximating the operator $\partial^2/(\partial\xi^i\partial\xi^j)$, by the central differences. The derivatives in J, a^{ij}, and $\boldsymbol{R}$ are also approximated by the central differences. The initial transformation $\boldsymbol{s}(\boldsymbol{\xi}, 0) = \boldsymbol{s}_0(\boldsymbol{\xi})$ is found through the formulas of transfinite interpolations.

An example of a three-dimensional prismatic spatial grid generated with the use of this scheme is demonstrated in Fig. 10.7.

10.3 Spectral Element Algorithm

The inverted diffusion equations in a divergent form (9.135) may be solved by a parallel code, using spectral elements for spatial discretization, Newton-Krylov methods for solution, and an adaptive time step.

Spatial discretization by high-order spectral elements is a method of exploiting the best features of both grid-based methods and global spectral representation. Grid-based methods, such as the finite difference approach described above, lead to nearest neighbor coupling and its resultant sparse matrix structure, and lends itself to parallelization by domain decomposition and that kind of adaptive gridding. On the other hand, convergence of the spatial truncation error is relatively slow, typically a low power of the grid spacing h. Global spectral methods overcome the latter problem, offering exponential convergence with increasing numbers of basis functions, but lead to large, dense matrices and offer no obvious way to use adaptive gridding and parallelization by domain decomposition. With spectral elements, there is a relatively coarse grid, and within each grid cell, there is a local expansion in basis functions based on orthogonal polynomials. The grid provides nearest-neighbor coupling while the spectral expansion provides exponential convergence.

All equations for spectral elements are to be expressed in flux-source form,

$$\frac{\partial u^k}{\partial t} + \nabla \cdot \mathbf{F}^k = S^k. \tag{10.39}$$

This very form has the following parabolic system:

$$\frac{\partial s^k}{\partial t} - \frac{\partial}{\partial \xi^j}(Jw(\mathbf{s})g_{\boldsymbol{\xi}}^{jk}) = 0, \quad j,k = 1,\ldots,n\ , \tag{10.40}$$

with identification $u = s,\ x^i = s^i$, obtained from grid equations (9.135) in the same manner as the system in (10.17) from the Eq. (10.11). The dependent variables u^k in (10.39) within each grid cell are expanded in a spectral basis $\alpha_j(x)$,

$$u^k(t,\mathbf{x}) \approx \sum_{j=0}^{n} u_j^k(t)\alpha_j(\mathbf{x}). \tag{10.41}$$

Spatially discretized equations are obtained through a Galerkin method, taking the scalar product of (10.39) with each basis function and integrating by parts to obtain

$$\ddot{M}\dot{\mathbf{u}} = \mathbf{r} \equiv \int_{X^n} (S^k\alpha_i + \mathbf{F}^k \cdot \nabla\alpha_i)d\mathbf{x} - \int_{\partial X^n} \mathbf{F}_i^k \cdot \hat{\mathbf{n}}d\mathbf{x}. \tag{10.42}$$

with $\ddot{M}$ the mass matrix, $M_{i,j} \equiv (\alpha_i, \alpha_j)$, and the $\mathbf{u}$ the vector of mode amplitudes $u_j^k(t)$. Integrals are evaluated by Gaussian quadrature to an order appropriate to the

degree of the Jacobi polynomials. Fluxes and sources may depend in an arbitrary nonlinear manner on t, $\mathbf{x}$, u^k, and ∇u^k. The code is structured in such a way that the details of discretization and the specification of physics equations are separated into different subroutines, making it as simple as possible to encode complex physics. The discretized flux-source form preserves conservation properties to high order. Elliptic equations are treated by zeroing the mass matrix.

Time discretization of (10.42) is fully implicit in order to treat multiple time scales efficiently and accurately,

$$\ddot{M}\left(\frac{\mathbf{u}^+ - \mathbf{u}^-}{h}\right) = \theta \mathbf{r}^+(\mathbf{u}^+) + (1-\theta)\mathbf{r}^-(\mathbf{u}^-) \tag{10.43}$$

with the time-centering parameter θ normally chosen as 1/2 (Crank-Nicholson) for accuracy. Solution of (10.43) requires finding the roots of the nonlinear residual,

$$\mathbf{R}(u^+) \equiv \ddot{M}(\mathbf{u}^+ - \mathbf{u}^-) - h[\theta \mathbf{r}^+ + (1-\theta)\mathbf{r}^-] = \mathbf{0}, \tag{10.44}$$

solved by Newton's iteration,

$$\mathbf{R} + \ddot{J}\delta\mathbf{u}^+ = \mathbf{0}, \quad \delta\mathbf{u}^+ = -\ddot{J}^{-1}\mathbf{R}(\mathbf{u}^+), \quad \mathbf{u}^+ \to \mathbf{u}^+ + \delta\mathbf{u}^+ \tag{10.45}$$

with the Jacobian defined as $\ddot{J} \equiv \ddot{M} - h\theta\{\partial r_i^+ / \partial u_j^+\}$.

Efficient solution of the large sparse linear system in (10.45) is greatly enhanced by the method of static condensation. Because of the C^0 nature of the spectral element representation, discussed above, higher-order elements in one grid cell couple to those in neighboring grid cells only through the shared linear finite elements which straddle cell boundaries. To solve a linear system $\ddot{A}\mathbf{x} = \mathbf{b}$, we partition the dependent variables into (1) element boundary terms and (2) element interior terms, for example, in two dimensions, the system is expressed in the form

$$\ddot{A}_{11}\mathbf{x}_1 + \ddot{A}_{12}\mathbf{x}_2 = \mathbf{b}_1, \tag{10.46}$$

$$\ddot{A}_{21}\mathbf{x}_1 + \ddot{A}_{22}\mathbf{x}_2 = \mathbf{b}_2. \tag{10.47}$$

Solving (10.47) for $\mathbf{x}_2$,

$$\ddot{A}_{22}\mathbf{x}_2 = \mathbf{b}_2 - \ddot{A}_{21}\mathbf{x}_1, \tag{10.48}$$

and substituting it into (10.46), we obtain an equation for the Shur complement,

$$(\ddot{A}_{11} - \ddot{A}_{12}\ddot{A}_{22}^{-1}\ddot{A}_{21})\mathbf{x}_1 = \mathbf{b}_1 - \ddot{A}_{12}\ddot{A}_{22}^{-1}\mathbf{b}_2. \tag{10.49}$$

Equation (10.48), involving the relatively small, dense, local matrix $\ddot{A}_{22}$, is solved locally using LAPACK routines. It parallelizes perfectly over grid cells, requiring no communication once $\mathbf{x}_2$ is determined. Equation (10.49), greatly condensed in size from the original system, is solved globally and iteratively by Krylov subspace

routine GMRES, using the PETSc library, preconditioned by additive Schwarz ILU factorization with substantial fill-in and overlap. The most efficient parallel operation is obtained with one grid cell per processor. This is feasible because the use of high-order spectral elements makes it possible to achieve good spatial resolution with relatively few grid cells.

For generating a numerical grid with node clustering in the zones of large values of a function $v(\mathbf{s})$, the measure of departure from the necessary grid can be expressed in the form

$$\sigma(\mathbf{s}) = Z[v](\mathbf{s}) g_{sx}^{kl} \frac{\partial \xi^i}{\partial s^k} \frac{\partial \xi^i}{\partial s^l}, \quad i,k,l = 1,\ldots,n \tag{10.50}$$

where $Z[v]$ is a positive operator such that the function $Z[v](\mathbf{s})$ is large (small) where $v(\mathbf{s})$ is small (large). This measure for generating adaptive grids in domains was introduced in Danaev et al. (1980) and Winslow (1981). Consequently, the contravariant elements of the control metric in S^{xn} are as follows:

$$g^{ij}(\mathbf{s}) = Z[v](\mathbf{s}) g_{sx}^{ij}, \quad i,j = 1,\ldots,n\,. \tag{10.51}$$

This contravariant metric tensor can also be used for providing node clustering in the zones of the large variation of a function $\mathbf{f}(\mathbf{s})$ by introducing for this purpose a function $v(grad\ \mathbf{f})$ such that v is large where $|grad\ \mathbf{f}|$ is large, and vice versa.

We choose in the control metric (10.51) the weight function $v(\mathbf{s})$ and assume $Z[v](\mathbf{s}) = 1/v(\mathbf{s})$, to reflect the spatial truncation error in the spectral element representation. In each grid cell $\bar{\Omega}$, we define the spatial truncation error as the ratio of the L^2 norm of the highest-order polynomial $\delta u(\mathbf{s})$ and that of the full solution $u(\mathbf{s})$, but because the spatial discretization error for spectral element methods is exponentially convergent with an increasing number of terms, we use the log of this norm,

$$\delta\bar{\Omega} \equiv \frac{1}{2}\left(\frac{\int\limits_{\bar{\Omega}} \delta u^2(\mathbf{s})\mathrm{d}\mathbf{s}}{\int\limits_{\bar{\Omega}} u^2(\mathbf{s})\mathrm{d}\mathbf{s}}\right). \tag{10.52}$$

Since this function is piecewise constant over each grid cell but we need a smooth function, we use a least-squares bicubic spline fit. Finally, in order to control the range of variation of v, we define

$$v(\mathbf{s}) = 1 + \alpha\left(\frac{\delta - \delta_{min}}{\delta_{max} - \delta_{min}}\right) \tag{10.53}$$

with α an adjustable constant. When $\delta = \delta_{min}$, $v(\mathbf{s}) = 1$, and when $\delta = \delta_{max}$, $v(\mathbf{s}) = 1 + \alpha$. Figure 10.8 (*left-hand*) shows the resulting grid lines obtained by solution of equations (10.40) with $w(\mathbf{s}) = 1/v(\mathbf{s})$. Note that the grid spacing is coarse where v is small and fine where v is large. Thus, the grid is refined where the spatial truncation error is large and rarefied where it is small. Figure 10.8 (*left-hand*) exhibits a grid for both alignment and adaptation and scaled grid density. Figure 10.8 (*center*) shows the

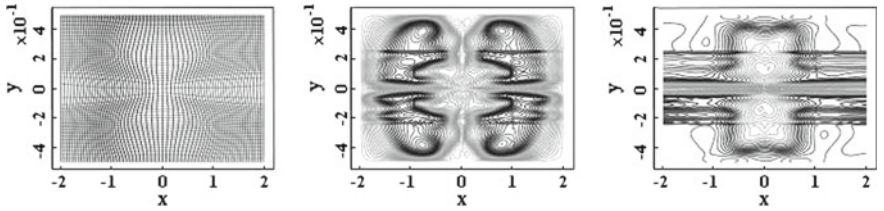

Fig. 10.8 Grid lines for both alignment and adaptation (*left*). Contour plot of alignment error for both alignment and adaptation (*center*). Density of grid lines for both alignment and adaptation (*right*)

resulting weight function for a magnetic reconnection problem. Figure 10.8 (*right-hand*) shows a contour plot of the inverse Jacobian of the transformation $\mathbf{s}(\boldsymbol{\xi})$, which may be interpreted as grid density. The pictures in Fig. 10.8 were formed by A. Glasser who used a spectral element method, developed by Glasser and Tang (2004), for computing plasmas and inverted diffusion grid equations for generating adaptive, field-aligned grids (see Glasser et al. (2005, 2006)).

10.4 Finite Element Method

The finite element method has diverse applications to problems in engineering and science. We demonstrate here its application to numerical grid generation by solution of problem (10.17) for $n = 2$ whose equations are written as

$$\frac{\partial s^l}{\partial t} - (J)^p\Big[g^{\xi}_{22}\frac{\partial^2 s^l}{\partial \xi^1 \partial \xi^1} - 2g^{\xi}_{12}\frac{\partial^2 s^l}{\partial \xi^1 \partial \xi^2} + g^{\xi}_{11}\frac{\partial^2 s^l}{\partial \xi^2 \partial \xi^2} - R^l(\boldsymbol{s})\Big] = 0\,, \quad l = 1, 2\,. \tag{10.54}$$

These equations are replaced by the following relations:

$$\int_{\Xi^2}\Big(\frac{\partial s^l}{\partial t}v^h - (J)^p\Big[(-1)^{i+j}g^{\xi}_{3-i3-j}\frac{\partial^2 s^l}{\partial \xi^i \partial \xi^j}v^h - R^l v^h\Big]\Big)\mathrm{d}\boldsymbol{\xi} = 0\,, \quad i, j, l = 1, 2\,, \tag{10.55}$$

where v^h are trial functions. Choosing a basis $\varphi_1, \ldots, \varphi_N$ for the trial functions at the interior grid nodes

$$\varphi_p(\boldsymbol{\xi}_k) = \begin{cases} 1, & k = p \\ 0, & k \neq p, \end{cases}$$

and another basis $\phi_{N+1}, \ldots, \phi_{N^r}$, at the boundary grid nodes,

$$\phi_p(\boldsymbol{\xi}_k) = \begin{cases} 1, & k = p \\ 0, & k \neq p, \end{cases}$$

where N^{Γ} is a number of interior and boundary grid nodes, we have an expansion of the functions $s^l(\boldsymbol{\xi}, t)$, $l = 1,\ 2$

$$s^l(\boldsymbol{\xi}, t) = s_p^{l\Gamma}\phi_p + s_k^l\varphi_k\ , \quad k = 1, \ldots, N, \quad p = N+1, \ldots, N^{\Gamma}\ , \quad l = 1, 2\ . \tag{10.56}$$

Therefore, from (10.55), we obtain the following system of equations:

$$\begin{aligned} &\frac{\partial s_k^l}{\partial t}\int\limits_{\Xi^2}\varphi_k\varphi_m \mathrm{d}\boldsymbol{\xi} = -s_k^l\int\limits_{\Xi^2}\frac{\partial}{\partial\xi^i}\Big((J)^p(-1)^{i+j}g^{\boldsymbol{\xi}}_{3-i3-j}\varphi_k\Big)\frac{\partial\varphi_m}{\partial\xi^j}\mathrm{d}\boldsymbol{\xi} \\ &-\int\limits_{\Xi^2}(J)^p R^l\varphi_m \mathrm{d}\boldsymbol{\xi}, \quad k, m = 1, \ldots, N, \quad i, j, l = 1, 2, \end{aligned} \tag{10.57}$$

or in a matrix form $\boldsymbol{M} = \{M_{mk}\}$, $\boldsymbol{K} = \{K_{mk}\}$, $\boldsymbol{F} = \{F_1, \ldots, F_m\}, k,\ m = 1, \ldots, N$

$$\boldsymbol{M}\frac{\partial \boldsymbol{s}^l}{\partial t} = \boldsymbol{K}\boldsymbol{s}^l - \boldsymbol{F}\ , \tag{10.58}$$

where

$$\begin{aligned} &M_{mk} = \int\limits_{\Xi^2}\varphi_k\varphi_m \mathrm{d}\boldsymbol{\xi}, \quad K_{mk} = -\int\limits_{\Xi^2}\frac{\partial}{\partial\xi^i}\Big((J)^p(-1)^{i+j}g^{\boldsymbol{\xi}}_{3-i3-j}\varphi_k\Big)\frac{\partial\varphi_m}{\partial\xi^j}\mathrm{d}\boldsymbol{\xi}, \\ &F_m = \int\limits_{\Xi^2}(J)^p R^l\varphi_m \mathrm{d}\boldsymbol{\xi}, \quad k, m = 1, \ldots, N, \quad i, j, l = 1, 2. \end{aligned} \tag{10.59}$$

Solving system (10.58) gives the values s_k^l, $l = 1, 2$, $k = 1, \ldots, N$, and consequently the values of the grid node coordinates. A more detailed description of the algorithm was originally published in Vaseva and Liseikin (2011).

Figures 10.9 and 10.10 illustrate an application of the finite element method to generation of adaptive triangle grids based on the solution of inverted diffusion equations.

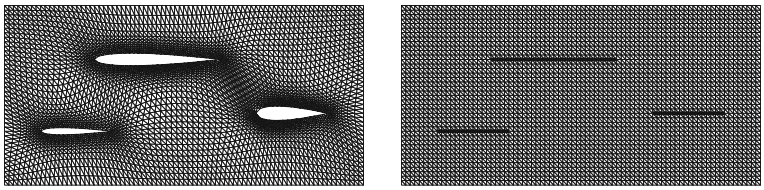

Fig. 10.9 Adaptive grid with node clustering near the boundaries of wings specified analytically (*left*) and the reference grid in Ξ^2 (*right*)

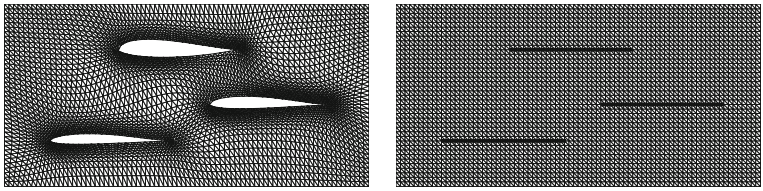

Fig. 10.10 Adaptive grid with node clustering near the boundaries of wings specified discretely (*left*) and the reference grid in Ξ^2 (*right*)

10.5 Inverse Matrix Method

Sections 10.1–10.4 consider algorithms for generating numerical grids by solving a matrix equation

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \quad \boldsymbol{A} = \{a_{ij}\}, \quad i, j = 1, \ldots, M, \tag{10.60}$$

without finding the inverse matrix $\boldsymbol{A}^{-1}$. This section describes an algorithm for finding the inverse matrix $\boldsymbol{A}^{-1}$ provided that there exists a nondegenerate matrix $\boldsymbol{A}(s)$ whose coefficients are dependent on a parameter s, $0 \le s \le 1$, i.e. $\boldsymbol{A}(s) = \{a_{ij}(s)\}$ and $\boldsymbol{A}(1) = \boldsymbol{A}$, while the inverse matrix $\boldsymbol{A}^{-1}(0)$ of the matrix $\boldsymbol{A}(0)$ is known. For example, if $\boldsymbol{A} = \{a_{ij}\}$ is some matrix with a diagonal domination, then $\boldsymbol{A}(s)$ may be defined as

$$\boldsymbol{A}(s) = (1-s)\boldsymbol{D} + s\boldsymbol{A}, \quad \text{i. e.} \quad a_{ij}(s) = (1-s)d_{ij} + sa_{ij}, \quad i, j = 1, \ldots, M, \tag{10.61}$$

where $\boldsymbol{D} = \{d_{ij}\}$ is the matrix whose elements equal zero if $i \ne j$, i.e. $d_{ij} = 0$, $i \ne j$, and its diagonal elements coincide with the diagonal elements of the matrix $\boldsymbol{A}$, i.e. $d_{ii} = a_{ii}$ for every fixed index $i = 1, \ldots, M$. Thus, $\boldsymbol{A}(0) = \boldsymbol{D}$, $\boldsymbol{D}^{-1} = \{b^{ij}\}$, $b^{ij} = 0$ if $i \ne j$, $b^{ii} = 1/a_{ii}$, i – fixed. Taking into account that $a_{ij}(s)b^{jk}(s) = \delta^i_j$, $i,\ j = 1, \ldots, M$, we have

$$\frac{\partial}{\partial a_{lp}}(a_{ij}b^{jk}) = \delta^l_i\delta^p_j b^{jk} + a_{ij}\frac{\partial b^{jk}}{\partial a_{lp}} = \delta^l_i b^{pk} + a_{ij}\frac{\partial b^{jk}}{\partial a_{lp}} = 0,\ i, j, k, l, p = 1, \ldots, M\,.$$

Multiplying these equations by b^{ti} and summing over i, we obtain

$$\frac{\partial b^{tk}(s)}{\partial a_{lp}(s)} = -b^{tl}(s)b^{pk}(s)\ , \quad k, l, p, t = 1, \ldots, M\ .$$

Therefore, for the elements of the inverse matrix $\boldsymbol{A}^{-1}(s) = \{b^{ij}(s)\}$, we obtain a system of ordinary nonlinear differential equations

$$\frac{\mathrm{d}}{\mathrm{d}s}b^{ij}(s) = \frac{\partial b^{ij}(s)}{\partial a_{kl}(s)}\frac{\mathrm{d}}{\mathrm{d}s}a_{kl}(s) = -b^{ik}(s)b^{lj}(s)\frac{\mathrm{d}}{\mathrm{d}s}a_{kl}(s)\ , \quad i,j,k,l = 1,\ldots,M, \quad 0 < s \le 1\ , \tag{10.62}$$

with the initial condition $b^{ij}(0)$, $i,\ j = 1,\ldots,M$, for $s = 0$. In particular, for the matrix $\{a_{ij}(s)\}$ defined by formula (10.61), we obtain an autonomous system of ordinary differential equations with the initial condition:

$$\frac{\mathrm{d}}{\mathrm{d}s}b^{ij}(s) = -b^{ik}(s)b^{lj}(s)(1-\delta^{kl})a_{kl}, \quad i,j,k,l = 1,\ldots,M, \quad 0 < s \le 1,$$
$$b^{ij}(0) = 0, \quad \text{if } i \ne j, \quad b^{ij}(0) = \frac{1}{a_{ij}}, \quad \text{if } i = j, \quad i,j = 1,\ldots,M\ .$$

Solving the initial problem for Eq. (10.62), for example, through the Runge Kutta method on the numerical grid $s_i = ih$, $i = 0,\ldots,N$, $h = 1/N$, we find for $s = 1$ approximate values of the elements of the matrix $\boldsymbol{A}^{-1}$. Then, an approximate solution of problem (10.60) is obtained as $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$.

Note that the original description of the method was given in Liseikin (2014a, b).

10.6 Method of Minimization of Energy Functional

This section describes another finite-difference grid generation algorithm based on the minimization of inverted energy functional (9.23). Following Charakch'yan and Ivanenko (1988, 1997), the algorithm is first expounded for the two-dimensional version of the functional in the Euler metric, i.e.

$$I_{IS}[\mathbf{s}] = \int\limits_{\Xi^2} \frac{(x_\xi)^2 + (x_\eta)^2 + (y_\xi)^2 + (y_\eta)^2}{J}\mathrm{d}\xi\mathrm{d}\eta, \tag{10.63}$$

where $J = x_\xi y_\eta - x_\eta y_\xi$, and then an explanation is given as to how it can be generalized to monitor metrics and other dimensions. Note that the functional (9.23) in the Euler metric $g^{\mathbf{s}}_{ij} = \delta^i_j$, $i,j = 1,2$, becomes the functional (10.63) when the following designations are assumed:

$$\xi^1 = \xi, \quad \xi^2 = \eta, \quad \mathbf{s}(\xi,\eta) = [x(\xi,\eta), y(\xi,\eta)].$$

By the algorithm, the functional (10.63) is approximated by a discrete functional $I^h[\ ]$. This is made by approximating the integrand in (10.63) at each grid cell of the logical domain Ξ^2 and then carrying out summation over all cells.

10.6.1 Generation of Fixed Grids

The problem of grid generation is treated as a discrete analog of the problem of finding the components $x(\xi, \eta)$ and $y(\xi, \eta)$ of the intermediate transformation $\mathbf{s}(\xi, \eta)$ producing one-to-one mapping of the logical square

$$0 < \xi < 1,\ 0 < \eta < 1$$

onto a physical domain X^2.

Instead of the logical square on the plane ξ, η, the parametric rectangle

$$1 < \xi < N,\ 1 < \eta < M.$$

is introduced to simplify the computational formulas. This rectangle is associated with the square grid (ξ_i, η_j) on the plane ξ, η such that $\xi_i = i,\ \eta_j = j,\ i = 1, \ldots, N;\ j = 1, \ldots, M$.

It is readily shown that if a smooth mapping of one domain onto another with a one-to-one transformation between boundaries possesses a positive Jacobian, then such a mapping will be one-to-one. Hence, the grid coordinate system, generated in the domain X^2, will be non-degenerate if the Jacobian of the mapping $\mathbf{s}(\xi, \eta) = [x(\xi, \eta), y(\xi, \eta)]$ is positive:

$$J = x_\xi y_\eta - x_\eta y_\xi > 0. \tag{10.64}$$

Thus, the problem of the construction of the grid coordinates in the domain X^2 can be formulated as the problem of finding a smooth mapping of the parametric rectangle onto the domain X^2, which satisfies the condition of the Jacobian positiveness.

Formulation of Discrete Functional

Let the coordinates $(x, y)_{ij}$ of grid nodes be given. To construct the mapping $x^h(\xi, \eta)$, $y^h(\xi, \eta)$ of the parametric rectangle onto the domain X^2 such that $x^h(i, j) = x_{ij}$ and $y^h(i, j) = y_{ij}$, quadrilateral isoparametric finite elements are used. The square cell numbered as $i + 1/2, j + 1/2$ on the plane ξ, η is mapped onto the quadrilateral cell on the plane x, y, formed by the nodes with coordinates $(x, y)_{ij}$, $(x, y)_{ij+1}$, $(x, y)_{i+1j+1}$, $(x, y)_{i+1j}$. The cell vertices are numbered from 1 to 4 in the clockwise direction. The node (i, j) corresponds to the vertex 1, node $(i, j + 1)$ to vertex 2, and so on. Each vertex is associated with a triangle: vertex 1 with $\triangle_{412}$, vertex 2 with $\triangle_{123}$, and so on. The doubled area $J_k,\ k = 1, 2, 3, 4$, of these triangles is introduced as follows:

$$J_k = (x_{k-1} - x_k)(y_{k+1} - y_k) - (y_{k-1} - y_k)(x_{k+1} - x_k)$$

where one should put $k - 1 = 4$ if $k = 1$, $k + 1 = 1$ if $k = 4$.

The functions x^h, y^h for $i \le \xi \le i+1,\ j \le \eta \le j+1$ are represented in the form

$$\begin{aligned} x^h(\xi,\eta) &= x_1 + (x_4 - x_1)(\xi - i) + (x_2 - x_1)(\eta - j) \\ &\quad +(x_3 - x_4 - x_2 + x_1)(\xi - i)(\eta - j), \\ y^h(\xi,\eta) &= y_1 + (y_4 - y_1)(\xi - i) + (y_2 - y_1)(\eta - j) \\ &\quad +(y_3 - y_4 - y_2 + y_1)(\xi - i)(\eta - j). \end{aligned} \tag{10.65}$$

Each side of the square is linearly transformed onto the appropriate side of the quadrilateral. Consequently, the global transformation x^h, y^h is continuous on the cell boundaries. To check the one-to-one property of the transformation (10.65), we write out the expression for its Jacobian

$$J^h = x^h_\xi y^h_\eta - x^h_\eta y^h_\xi = \det \begin{pmatrix} x_4 - x_1 + A(\eta - j) & x_2 - x_1 + A(\xi - i) \\ y_4 - y_1 + B(\eta - j) & y_2 - y_1 + B(\xi - i) \end{pmatrix},$$

where $A = x_3 - x_4 - x_2 + x_1$, $B = y_3 - y_4 - y_2 + y_1$. The function J^h is linear, not bilinear, since the coefficient before $\xi\eta$ in this determinant is equal to zero. Consequently, if $J^h > 0$ at all corner points of the square, it does not vanish inside this square. At the corner node 1 ($\xi = i,\ \eta = j$) of the cell $i + 1/2$, $j + 1/2$, the Jacobian equals

$$J^h(i,j) = (x_4 - x_1)(y_2 - y_1) - (y_4 - y_1)(x_2 - x_1),$$

i.e. $J^h(i,j) = J_1$ is the doubled area of the triangle $\triangle_{412}$, introduced above. From this follows that the condition of the Jacobian positiveness $x^h_\xi y^h_\eta - x^h_\eta y^h_\xi > 0$ is equivalent to the system of inequalities

$$[J_k]_{i+1/2 j+1/2} > 0,\ \ k = 1,2,3,4;\ i = 1,\dots,N-1;\ j = 1,\dots,M-1. \tag{10.66}$$

If conditions (10.66) are satisfied, then all the grid cells are convex quadrilaterals. The set of grids satisfying these inequalities is called a convex grid set and denoted by D. This set belongs to the Euclidean space R^{N_1}, where $N_1 = 2(N-2)(M-2)$ is the total number of degrees of freedom of the grid equal to twice the number of its interior nodes.

Finally, the problem is formulated as follows. The convex grid, satisfying inequalities (10.66), must be generated in the domain X^2 for the given coordinates of the boundary nodes.

The mapping $x(\xi,\eta)$, $y(\xi,\eta)$ is approximated by functions $x^h(\xi,\eta)$, $y^h(\xi,\eta)$ introduced in (10.65). Substituting those expressions in (10.63) and replacing integrals over square cells by the quadrature formulas with nodes coinciding with the grid vertices on the plane ξ,η, the following discrete analog of the functional (10.63) is obtained:

$$I^h = \sum_{i=1}^{N-1} \sum_{j=1}^{M-1} \sum_{k=1}^{4} \frac{1}{4} [F_k]_{i+1/2\ j+1/2}, \tag{10.67}$$

where F_k is the integrand evaluated in the k - th grid node as

$$F_k = [(x_{k+1} - x_k)^2 + (x_k - x_{k-1})^2 + (y_{k+1} - y_k)^2 + (y_k - y_{k-1})^2]J_k^{-1}, \quad (10.68)$$

and J_k is the doubled area of the triangle introduced above.

Notice some properties of the function (10.67). For this purpose, we introduce a parametric rectangle $0 < \xi < 1, 0 < \eta < \alpha$, where $\alpha = (M-1)/(N-1)$ is the constant, instead of the unit logical square as a domain of integration in (10.63). In this case, the continuous limit of the expression $I^h/(N-1)^2$ when $N, M \to \infty$ in such a way, that $(M-1)/(N-1) = \alpha = const$, will be the functional (10.63).

The following identity is readily obtained:

$$\begin{aligned} I &= \int_0^1 \int_0^\alpha \frac{x_\xi^2 + y_\xi^2 + x_\eta^2 + y_\eta^2 - 2(x_\xi y_\eta - x_\eta y_\xi) + 2(x_\xi y_\eta - x_\eta y_\xi)}{J} d\xi d\eta \\ &= \int_0^1 \int_0^\alpha \frac{(x_\xi - y_\eta)^2 + (x_\eta - y_\xi)^2}{J} d\xi d\eta + 2\alpha. \end{aligned}$$

From this follows that the functional (10.63) has a lower bound equal to 2α. If this minimum is attained, the mapping $\mathbf{s}(\xi, \eta)$ is conformal:

$$x_\xi = y_\eta, \quad x_\eta = -y_\xi.$$

To find out the corresponding property of discrete analog (10.67) of functional (10.63), let us consider one term in (10.68) for $k = 2$. We can assume that $x_2 = 0$ and $y_2 = 0$, since expression (10.68) contains only finite differences of the grid node coordinates. In this case, we obtain the following identity:

$$\begin{aligned} F_2 &= \frac{x_1^2 + y_1^2 + x_3^2 + y_3^2}{x_1 y_3 - x_3 y_1} \\ &= \frac{x_1^2 + y_1^2 + x_3^2 + y_3^2 - 2(x_1 y_3 - x_3 y_1) + 2(x_1 y_3 - x_3 y_1)}{x_1 y_3 - x_3 y_1} \\ &= \frac{(x_1 - y_3)^2 + (x_3 + y_1)^2}{x_1 y_3 - x_3 y_1} + 2. \end{aligned}$$

From this follows that the function $I^h/(N-1)^2$ has on the set D a lower bound equal to $2(M-1)/(N-1)$. If this minimum is attained, the coordinates of the grid nodes satisfy a discrete analog of the conformal conditions

$$x_1 = y_3, \quad x_3 = -y_1.$$

If these conditions are satisfied for all cells, every grid cell will be a square.

Note that the function (10.67) is not convex and, in principle, multiple solutions may exist.

The function I^h also possesses the following very important property. If $G \to \partial D$ for $G \in D$, where ∂D is the boundary of the set of convex grids D, i.e. if at least one of the quantities J_k tends to zero for some cell while remaining positive, then $I^h(G) \to +\infty$. In fact, suppose that $J_k \to 0$ in (10.68) for some cell, but I^h does not tend to $+\infty$. Then, the numerator in (10.68) must also tend to zero, i.e. the lengths of two sides of the cell tend to zero. Consequently, the areas of all triangles that contain these sides must also tend to zero. Repeating the argument as many times as necessary, we conclude that the lengths of the sides of all grid cells, including those at the boundary of the domain, must tend to zero, which is impossible.

Thus, if the set D is not empty, the system of algebraic equations

$$R_x = \frac{\partial I^h}{\partial x_{ij}} = 0, \quad R_y = \frac{\partial I^h}{\partial y_{ij}} = 0, \quad i = 2, \ldots, N-1; \quad j = 2, \ldots, M-1, \tag{10.69}$$

has at least one solution which is a convex grid. To find it, one must first find a certain initial grid $G_0 \in D$, and then use some method of unconstrained minimization. Since the function (10.67) has the infinite barrier on the boundary of the set D, each step of the method can be chosen so that the grid always remains convex. Note that in the common case, the discrete grid-generation Eq. (10.69) may have multiple solutions, but numerical experiments have not met with such opportunity.

Method of Minimization

First, we consider a method for minimizing the function (10.67) assuming that the initial grid $G_0 \in D$ has been found. Suppose the grid at the l-th step of the iterations is determined. For finding the grid nodes at the $(l+1)$-th step, the quasi-Newtonian procedure for each interior node can be used:

$$\begin{aligned} \tau R_x + \frac{\partial R_x}{\partial x_{ij}}(x_{ij}^{l+1} - x_{ij}^{l}) + \frac{\partial R_x}{\partial y_{ij}}(y_{ij}^{l+1} - y_{ij}^{l}) = 0, \\ \tau R_y + \frac{\partial R_y}{\partial x_{ij}}(x_{ij}^{l+1} - x_{ij}^{l}) + \frac{\partial R_y}{\partial y_{ij}}(y_{ij}^{l+1} - y_{ij}^{l}) = 0 \end{aligned} \tag{10.70}$$

where τ is the iteration parameter. Note that (10.70) is not the Newton-Raphson iteration, because only a part of the second derivatives of (10.67) is taken into account. The rate of convergence for (10.70) is low by comparison. At the same time, the Newton-Raphson method gives us a much more complex system of linear equations at each iteration.

Each of the derivatives in (10.70) is the sum of twelve terms, in accordance with the number of triangles containing the given node as a vertex. Rather than write out such cumbersome expressions, the first and second derivatives of the terms in (10.67) are considered:

$$\frac{\partial F_k}{\partial x_{k-1}} = 2\frac{x_{k-1} - x_k}{J_k} - F_k \frac{y_{k+1} - y_k}{J_k}, \tag{10.71}$$

and so on. Arrays storing the derivatives of the function (10.67) were first cleared, and then all grid triangles were scanned and the appropriate derivatives added to the relevant elements of the arrays.

Now, an algorithm is suggested for the choice of the iteration parameter τ in (10.70), which was used only for the problems with moving boundaries. Recall that the minimized function (10.67) has the infinite barrier on the boundary of the set of convex grids D. Since the initial grid G_0 is convex, the iteration (10.70) gives, as a rule, a convex grid for any $\tau < 1$. But in extreme cases when G_0 is very close to the boundary of the set D, the grid $G(\tau)$ can cross the boundary of the set in the first iterations (10.70). Clearly, such a condition is fatal for the method because the same barrier on the boundary of the set D does not allow the iterations to return into the set D in the following iterations. To avoid this, a certain basic parameter τ_0 is chosen so that $G(\tau_0/2) \in D$ and $G(\tau_0) \in D$. In the beginning, $\tau_0 = 1$. If the above-mentioned conditions are violated, we put $\tau_0 = 1/4$ or $\tau_0 = 1/2$, depending on whether the grids $G(\tau_0/2)$ or $G(\tau_0)$ leave the set D, and so on.

In fixed boundary problems, the simple choice $\tau = \text{const} \cdot \tau_0$ is used. For time-dependent problems with moving boundaries, a version of the method of parabolas was developed. As the controlling quantity, the squared residual of the Eq. (10.70)

$$W = \sum_{i,j} (R_x^2 + R_y^2)_{i,j}$$

was used. The parabola $W(\tau)$ is constructed from the grids obtained for $\tau = 0$, $\tau = \tau_0/2$ and $\tau = \tau_0$. The parameter τ is then chosen so that $W(\tau) = \min$ in the interval $\theta\tau \le \tau \le \alpha\tau_0$. The parameter $\theta \sim 0.1$ is given a priori and bounds the value of τ away from zero. The parameter α bounds τ above, i.e. prevents a very large extrapolation along the parabola. If $\tau_0 = 1$, i.e. if the boundary of the set D is not crossed, we put $\alpha = 2$. If $\tau_0 < 1$, then $\alpha = 1$. Finally, if the algorithm gives $\tau < \tau_0/2$, the condition $I^h(\tau_0/2) < I^h(0)$ is checked. In the cases when this condition is found to be valid, $\tau = \tau_0/2$ was used.

For one iteration of the above method, a measurement of the computational cost gives the value of about double (but not three times) the cost of the simple iteration. The reason is that the second derivatives of the function (10.67) are not used in calculating W, while they are used in (10.70) to calculate the direction of minimization.

The algorithm described can be used only if the initial grid is convex. Otherwise, it is necessary either to obtain a convex grid through another algorithm as a preliminary stage of the method or to modify the computational formulas. The first approach is based on the minimization of the following function:

$$I_D = \sum_{i=1}^{N-1} \sum_{j=1}^{M-1} \sum_{k=1}^{4} \left([\epsilon - J_k]_{i+1/2\ j+1/2} \right)_+^2, \quad (f)_+ = \max(0, f), \tag{10.72}$$

for some given $\epsilon > 0$. This is accomplished through the gradient method with a suitable choice of the iteration parameter. The iterative process is broken off as soon

as all inequalities (10.66) are satisfied. This method was used by Charakch'yan (1993, 1994) for studying gas dynamics problems with moving boundaries when the initial interior grid nodes for minimizing (10.72) were taken from the previous time step. As a result, the initial grid is either convex or such that a convex grid is obtained after a few iterations.

In fixed boundary problems, the starting grid may be non-convex, containing numerous self-intersecting cells. In such a case the preliminary stage of the method based on minimizing (10.72) can be unsuitable. Therefore, another approach had been developed by Ivanenko (1988). The computational formulas (10.70) were modified so that the initial grid need not belong to the set D of convex grids. The quantities J_k appearing in the expressions for R_x, R_y and their derivatives are replaced with new quantities $\tilde{J}_k$

$$\tilde{J}_k = \begin{cases} J_k & if\ J_k > \epsilon, \\ \epsilon & if\ J_k \le \epsilon, \end{cases}$$

where $\epsilon > 0$ is some sufficiently small quantity.

It is quite important to choose an optimal value of ϵ so that the convex grid is constructed as quickly as possible. The method used for specifying the value of ϵ is based on the computation of the absolute value of the average area of triangles with negative areas

$$\epsilon = \max[\alpha S/(N + 0.01), \epsilon_1],$$

where S is twice the absolute value of the total area of triangles with negative areas, and N the number of these triangles. The quantity $\epsilon_1 > 0$ sets a lower bound on ϵ to avoid very large values appearing in the computations. The coefficient α is chosen experimentally and is in the range $0.3 \le \alpha \le 0.7$.

In practical implementation, an arbitrary set of grid nodes can be marked as movable during iterations, while all other nodes are considered as stationary. All the terms in the function (10.67) which become independent on movable nodes are excluded from computations. Since the boundary nodes are always marked as stationary, four terms in (10.67) corresponding to "corner" triangles $\{(1, 2); (1, 1); (2, 1)\}$, $\{(N - 1, 1); (N, 1); (N, 2)\}$, $\{(1, M - 1); (1, M); (2, M)\}$, and $\{(N - 1, M); (N, M); (N, M - 1)\}$ are always excluded from computations. As a result, the method becomes applicable to those domains for which the angle between two intersecting boundaries is greater than or equal to π, despite the fact that the corresponding grid cell becomes non-convex regardless of the positions of interior nodes.

Examples of the grids generated by this method are exhibited in Figs. 10.11 and 10.12. Figure 10.12 demonstrates the application of the algorithm to generation of a grid for computing a high-velocity impact of a thin foil (a) upon a conical target CD Lomonosov et al. (1997).

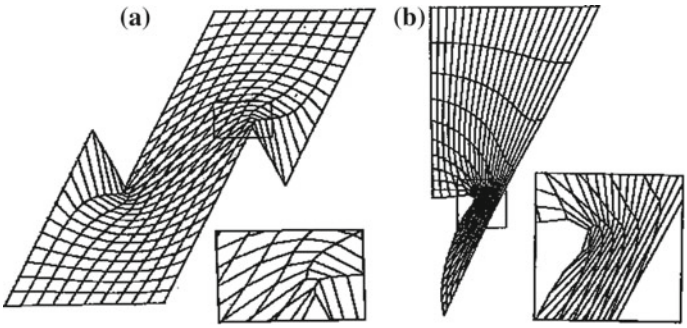

Fig. 10.11 Grids in a model domain (**a**) and for computing a cumulative jet (**b**)

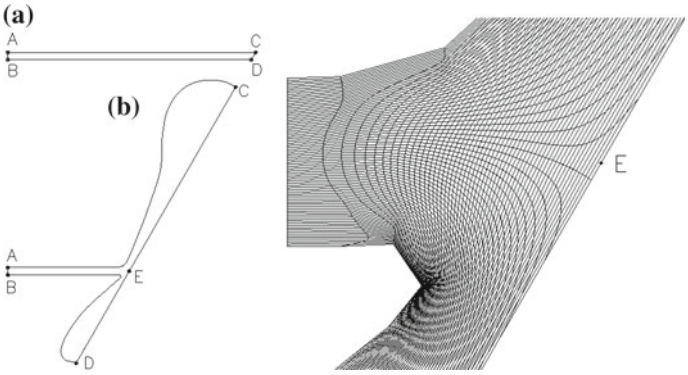

Fig. 10.12 A fragment of the grid (*right*) in the vicinity of the point E (*left*)

10.6.2 Adaptive Grid Generation

Numerical Algorithm

One approach to adaptive grid generation is based on the minimization of the functional (9.23) in the metric of a monitor surface.

Let the monitor surface be defined by a function $z = f(x, y)$ where $f \in C^1$. The expressions for the covariant elements and Jacobian of the monitor metric in the grid coordinates $\xi = \xi^1,\ \eta = \xi^2$ are as follows:

$$
\begin{aligned}
g^{\xi}_{11} &= g^{s}_{11}\Big(\frac{\partial x}{\partial \xi}\Big)^2 + 2g^{s}_{12}\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi} + g^{s}_{22}\Big(\frac{\partial y}{\partial \xi}\Big)^2,\\
g^{\xi}_{22} &= g^{s}_{11}\Big(\frac{\partial x}{\partial \eta}\Big)^2 + 2g^{s}_{12}\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \eta} + g^{s}_{22}\Big(\frac{\partial y}{\partial \eta}\Big)^2,\\
g^{\xi} &= (J)^2 g^{s} = (J)^2[1 + (f_x)^2 + (f_y)^2],
\end{aligned}
$$

where

$$
g^{s}_{11} = 1 + (f_x)^2, \quad g^{s}_{12} = f_x f_y, \quad g^{s}_{22} = 1 + (f_y)^2.
$$

Then, functional (9.23) for $n = 2$, with the identification $X^2 = S^2$, $\xi = \xi^1$, $\eta = \xi^2$, has the following form:

$$I_a[\boldsymbol{x}] = \int_0^1 \int_0^1 \frac{(x_\xi^2 + x_\eta^2)[1 + (f_x)^2] + (y_\xi^2 + y_\eta^2)[1 + (f_y)^2] + 2 f_x f_y (x_\xi y_\xi + x_\eta y_\eta)}{J[1 + (f_x)^2 + (f_y)^2]^{1/2}} d\xi d\eta. \tag{10.73}$$

Now we again consider the grid $(x, y)_{ij}$, $i = 1, \ldots, N$; $j = 1, \ldots, M$ and, to simplify the computational formulas, the parametric rectangle $1 < \xi < N$, $1 < \eta < M$ substitutes for the unit square $0 < \xi < 1$, $0 < \eta < 1$. The functional I_a is approximated by the function

$$I_a^h = \sum_{i=1}^{N-1} \sum_{j=1}^{M-1} \sum_{k=1}^{4} \frac{1}{4} [F_k]_{i+1/2\ j+1/2}, \tag{10.74}$$

$$F_k = \frac{D_1[1 + (f_x)_k^2] + D_2[1 + (f_y)_k^2] + 2D_3 (f_x)_k (f_y)_k}{J_k[1 + (f_x)_k^2 + (f_y)_k^2]^{1/2}}, \tag{10.75}$$

where

$$\begin{aligned} D_1 &= (x_{k-1} - x_k)^2 + (x_{k+1} - x_k)^2, \\ D_2 &= (y_{k-1} - y_k)^2 + (y_{k+1} - y_k)^2, \\ D_3 &= (x_{k-1} - x_k)(y_{k-1} - y_k) + (x_{k+1} - x_k)(y_{k+1} - y_k), \\ J_k &= (x_{k-1} - x_k)(y_{k+1} - y_k) - (x_{k+1} - x_k)(y_{k-1} - y_k). \end{aligned}$$

Derivatives $(f_x)_k$ and $(f_y)_k$ in the k-th cell vertex are equal to the corresponding values of derivatives, evaluated at the grid node ij

$$\begin{aligned} (f_x)_{ij} &= \frac{(f_{i+1j} - f_{i-1j})(y_{ij+1} - y_{ij-1}) - (f_{ij+1} - f_{ij-1})(y_{i+1j} - y_{i-1j})}{(x_{i+1j} - x_{i-1j})(y_{ij+1} - y_{ij-1}) - (x_{ij+1} - x_{ij-1})(y_{i+1j} - y_{i-1j})}, \\ (f_y)_{ij} &= \frac{(f_{i+1j} - f_{i-1j})(x_{ij+1} - x_{ij-1}) - (f_{ij+1} - f_{ij-1})(x_{i+1j} - x_{i-1j})}{(x_{i+1j} - x_{i-1j})(y_{ij+1} - y_{ij-1}) - (x_{ij+1} - x_{ij-1})(y_{i+1j} - y_{i-1j})}. \end{aligned} \tag{10.76}$$

These formulas must be modified for the boundary nodes. Indices "leaving" the computational domain must be replaced by the nearest boundary indices. For example, if $j = 1$, then $(i, j - 1)$ must be replaced by (i, j).

Function (10.74) possesses the same property as the function (10.67): $I_a^h(G) \to +\infty$ if $G \to \partial D$ for $G \in D$ where D is the set of convex grids, and ∂D is the boundary of the set.

As before, Eq. (10.70) are used to minimize the function I_a^h. Quantities $(f_x)_{ij}$ and $(f_y)_{ij}$ are assumed to be parameters, and therefore all their derivatives in (10.70) vanish. Note that if $(f_x)_{ij}$ and $(f_y)_{ij}$ vanish, the function I_a^h reduces to the function I^h (10.67).

The adaptive grid generation algorithm is formulated as follows:

1. Generate a grid for the given domain using the unconstrained minimization algorithm described.

2. Compute the values of the control function at each grid node. The result is f_{ij}.

3. Evaluate derivatives $(f_x)_{ij}$ and $(f_y)_{ij}$ using the formulas (10.76).

4. Make one step in the minimization process for the function I_a^h using Eq. (10.70) and compute new values of x_{ij} and y_{ij}.

5. Repeat starting with Step 2 to convergency.

It is important that at each step of the iterative process the grid remains convex.

Redistribution of Boundary Nodes

There are several ways to redistribute the grid nodes along the boundary ∂X^2 of the domain X^2 during adaptation. The simplest one is a fixed position of every point on ∂X^2, referred to as the "fixed position." However, if some physical quantities are not smooth (e.g. shock waves), then some instability in the mesh generation and, consequently, in the physical problem solution near the points where the discontinuity joins ∂X^2 may arise. In some methods, referred to as "unconstrained minimization", the boundary nodes are treated as interior and the vectors of their shift are projected onto ∂X^2. This method can be used only if the discontinuity is nearly orthogonal to ∂X^2. If not, then, when condensing, the boundary nodes overlap, adjacent cells degenerate, and modeling breaks. The next method, referred to as "1-D minimization", relies on using the 1-D functional along ∂X^2. This method is more robust than the two ones discussed above and can usually be used for adaptation. However, the 1-D and 2-D functionals are, as a rule, inconsistent. For this reason, the parameters of adaptation for the interior and boundary nodes should be selected separately. It requires additional work when modeling unsteady flow problems.

In the method suggested by Azarenok (2002), instead of (10.74), the function

$$\tilde{I}_a^h = \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\sum_{k=1}^{4}\frac{1}{4}[F_k]_{i+1/2j+1/2} + \sum_{ij\in\mathcal{L}}\lambda_{ij}G_{ij} = I_a^h + \sum_{ij\in\mathcal{L}}\lambda_{ij}G_{ij}, \qquad (10.77)$$

was minimized where the constraints $G_{ij} = G(x_{ij}\,,\ y_{ij}) = 0$ define ∂X^2, λ_{ij} are the Lagrange multipliers, and $\mathcal{L}$ is the set of the boundary nodes. The function $G(x, y)$ is assumed to be piecewise differentiable, so the function $\tilde{I}_a^h$ holds the infinite barrier on the boundary of the set of convex grids as I_a^h does if $f \in C^1$.

If the set of convex grids is not empty, the system of algebraic equations

$$R_x = \frac{\partial I_a^h}{\partial x_{ij}} + \lambda_{ij}\frac{\partial G_{ij}}{\partial x_{ij}} = 0, \quad R_y = \frac{\partial I_a^h}{\partial y_{ij}} + \lambda_{ij}\frac{\partial G_{ij}}{\partial y_{ij}} = 0, \quad G_{ij} = 0, \qquad (10.78)$$

has at least one solution that is a convex mesh. Here, $\lambda_{ij} = 0$ if $ij \notin \mathcal{L}$ and the constraints are defined for the boundary nodes $ij \in \mathcal{L}$.

Consider the method of minimizing the function (10.77) assuming the grid to be convex at the lth step of the iterative procedure. The quasi-Newton procedure for finding, the coordinates x_{ij}^{l+1}, y_{ij}^{l+1} from the system (10.78) was used:

$$\begin{aligned}
&\tau R_x + \frac{\partial R_x}{\partial x_{ij}}\,(x_{ij}^{l+1} - x_{ij}^{l}) + \frac{\partial R_x}{\partial y_{ij}}\,(y_{ij}^{l+1} - y_{ij}^{l}) + \frac{\partial R_x}{\partial \lambda_{ij}}\,(\lambda_{ij}^{l+1} - \lambda_{ij}^{l}) = 0,\\
&\tau R_y + \frac{\partial R_y}{\partial x_{ij}}\,(x_{ij}^{l+1} - x_{ij}^{l}) + \frac{\partial R_y}{\partial y_{ij}}\,(y_{ij}^{l+1} - y_{ij}^{l}) + \frac{\partial R_y}{\partial \lambda_{ij}}\,(\lambda_{ij}^{l+1} - \lambda_{ij}^{l}) = 0,\\
&\tau G_{ij} + \frac{\partial G_{ij}}{\partial x_{ij}}\,(x_{ij}^{l+1} - x_{ij}^{l}) + \frac{\partial G_{ij}}{\partial y_{ij}}\,(y_{ij}^{l+1} - y_{ij}^{l}) = 0,
\end{aligned}$$

where

$$\begin{aligned}
&\frac{\partial R_x}{\partial x_{ij}} = \frac{\partial^2 I_a^h}{\partial x_{ij}^2} + \lambda_{ij}\frac{\partial^2 G_{ij}}{\partial x_{ij}^2}, \quad \frac{\partial R_x}{\partial y_{ij}} = \frac{\partial^2 I_a^h}{\partial x_{ij}\partial y_{ij}} + \lambda_{ij}\frac{\partial^2 G_{ij}}{\partial x_{ij}\partial y_{ij}},\\
&\frac{\partial R_y}{\partial x_{ij}} = \frac{\partial^2 I_a^h}{\partial x_{ij}\partial y_{ij}} + \lambda_{ij}\frac{\partial^2 G_{ij}}{\partial x_{ij}\partial y_{ij}}, \quad \frac{\partial R_y}{\partial y_{ij}} = \frac{\partial^2 I_a^h}{\partial y_{ij}^2} + \lambda_{ij}\frac{\partial^2 G_{ij}}{\partial y_{ij}^2},\\
&\frac{\partial R_x}{\partial \lambda_{ij}} = \frac{\partial G_{ij}}{\partial x_{ij}}, \quad \frac{\partial R_y}{\partial \lambda_{ij}} = \frac{\partial G_{ij}}{\partial y_{ij}}.
\end{aligned}$$

Resolving the last equation of (10.79) with respect to $y_{ij}^{l+1} - y_{ij}^{l}$ and substituting it in the two remaining equations, the system

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_{ij}^{l+1} - x_{ij}^{l} \\ \lambda_{ij}^{l+1} - \lambda_{ij}^{l} \end{pmatrix} = \begin{pmatrix} a_{13} \\ a_{23} \end{pmatrix},$$

is obtained, where

$$\begin{aligned}
a_{11} &= \frac{\partial R_x}{\partial x_{ij}} - \frac{\partial R_x}{\partial y_{ij}}\,\frac{\partial G_{ij}}{\partial x_{ij}} \Big/ \frac{\partial G_{ij}}{\partial y_{ij}},\\
a_{12} &= \frac{\partial G_{ij}}{\partial x_{ij}},\\
a_{13} &= \tau\left[\frac{\partial R_x}{\partial y_{ij}}\,G_{ij} \Big/ \frac{\partial G_{ij}}{\partial y_{ij}} - R_x\right],\\
a_{21} &= \frac{\partial R_y}{\partial x_{ij}} - \frac{\partial R_y}{\partial y_{ij}}\,\frac{\partial G_{ij}}{\partial x_{ij}} \Big/ \frac{\partial G_{ij}}{\partial y_{ij}},\\
a_{22} &= \frac{\partial G_{ij}}{\partial y_{ij}},\\
a_{23} &= \tau\left[\frac{\partial R_y}{\partial y_{ij}}\,G_{ij} \Big/ \frac{\partial G_{ij}}{\partial y_{ij}} - R_y\right].
\end{aligned}$$

Denoting $\triangle = a_{11}a_{22} - a_{12}a_{21}$, $\triangle_1 = a_{13}a_{22} - a_{23}a_{12}$, $\triangle_2 = a_{11}a_{23} - a_{21}a_{13}$ (since $G_{ij} = 0$, the terms a_{13}, a_{23} are simplified), we obtain

$$x_{ij}^{l+1} = x_{ij}^{l} + \triangle_1/\triangle, \quad \lambda_{ij}^{l+1} = \lambda_{ij}^{l} + \triangle_2/\triangle, \tag{10.79}$$

while y_{ij}^{l+1} is determined from the third equation of (10.79). If the constraints are resolved in y in the form $G(x, y) = y - g(x) = 0$, then

$$\frac{\partial G_{ij}}{\partial x_{ij}} = -\frac{\partial g_{ij}}{\partial x_{ij}}, \quad \frac{\partial G_{ij}}{\partial y_{ij}} = 1,$$

and the upper formulas are simplified. Analogously, the constrains may be resolved in x in the form $G(x, y) = x - \tilde{g}(y) = 0$. Note that the equation $G(x, y) = 0$ can be locally resolved by one of these two forms.

If ∂X^2 is specified by parametric functions $x = x(t)$, $y = y(t)$ or tabular values $(x, y)_{ij}$, the following algorithm can be used. Assume the index j is fixed and i is variable. When calculating the coordinates of the (ij)th node, in the interval $(x_{i-1\,j}, x_{i+1\,j})$, we construct an interpolating parabola $t = t(x)$ using the values in three nodes $(i-1\,j)$, (ij), and $(i+1\,j)$. From (10.79), we compute an intermediate value $\tilde{x}_{ij}^{l+1}$; further from the interpolation formula, we determine $t_{ij} = t(\tilde{x}_{ij}^{l+1})$ and final values x_{ij}^{l+1}, y_{ij}^{l+1} from the parametric formulas.

Another way for redistributing the nodes along ∂X^2, given as parametric functions or by tabular values, employs an unconstrained minimization of the function in a parametric form and is based on solving the following system of algebraic equations, referred to as "parametric minimization":

$$R_t = R_x \frac{\partial x_{ij}}{\partial t_{ij}} + R_y \frac{\partial y_{ij}}{\partial t_{ij}} = 0,$$

via the quasi-Newton procedure

$$\tau R_t + \frac{\partial R_t}{\partial t_{ij}} (t_{ij}^{l+1} - t_{ij}^{l}) = 0. \tag{10.80}$$

Here,

$$\frac{\partial R_t}{\partial t_{ij}} = \frac{\partial R_x}{\partial x_{ij}} \left(\frac{\partial x_{ij}}{\partial t_{ij}}\right)^2 + \frac{\partial R_y}{\partial y_{ij}} \left(\frac{\partial y_{ij}}{\partial t_{ij}}\right)^2 + \left(\frac{\partial R_x}{\partial y_{ij}} + \frac{\partial R_y}{\partial x_{ij}}\right) \frac{\partial x_{ij}}{\partial t_{ij}} \frac{\partial y_{ij}}{\partial t_{ij}}$$
$$+ R_x \frac{\partial^2 x_{ij}}{\partial t_{ij}^2} + R_y \frac{\partial^2 y_{ij}}{\partial t_{ij}^2}, \qquad R_x = \frac{\partial I^h}{\partial x_{ij}}, \quad R_y = \frac{\partial I^h}{\partial y_{ij}}.$$

To the analytical control functions, both the constrained and parametric minimization give similar results. Real-world 2-D flow computations have shown that it is better to perform adaptation along the boundary using constrained minimization (10.79), since the procedure (10.80) may not ensure consistent redistribution of the nodes in X^2 and on ∂X^2.

The use of the constrained minimization without adaptation (i.e. when $f = const.$) means that we seek the conformal mapping $x(\xi, \eta)$, $y(\xi, \eta)$ of the parametric rec-

tangle onto the domain X^2 with an additional parameter, the so-called conformal modulus.

10.7 Parallel Mesh Generation

Parallel computing is an efficient tool for handling large multidimensional problems by distributing the computational effort and/or the memory requirements over the different computers available.

As to the mesh generation step, one parallelization approach consists of constructing the mesh in parallel by means of using a meshing method under interest which is to be updated in order to incorporate some degree of parallelism. Many classical methods for mesh generation are amenable to being performed in parallel, in particular, the Delaunay and quad-octree methods and the mapping methods based on the numerical solution of elliptic and parabolic equations such as the inverted Beltrami and diffusion equations.

The second approach to the parallelization of the meshing process consists of partitioning the domain by means of sub-domains, whose union forms a covering-up of the entire domain, prior to dispatching these to different processors, each of them generating a mesh on one sub-domain. Different classes of domain partition are encountered. Among these, some are based on graph partitions and some are purely geometric methods directly based on mesh partitions. All these methods apply to finite element type meshes, since a vicinity graph can be constructed based on the connections between the elements in a given mesh.

The partition of the domain as well as constructing the corresponding sub-meshes can be achieved either through a posteriori or a priori partitioning methods. The posteriori processing starts from the data of a large size fine mesh of the entire domain and then splits it into sub-meshes, while a priori processing uses the domain itself or a coarse mesh of it which is split into sub-domains.

For the priori processing, first, a partition of the domain is created. This step may start from the domain geometry or a reasonable coarse mesh of the entire domain. Once this partition is available, some sub-domains are then identified and meshed, each on one processor, thus taking advantage of the parallel capabilities of the computers right from the meshing stage. The global mesh is then achieved by merging all of the local meshes. The interface between two sub-domains is constructed either from the data of the coarse mesh or from the data of the domain boundary discretization. The meshes can also be constructed by using the meshes of the surfaces that constitute the interfaces between the sub-domains extracted from the given boundary mesh. This approach leads to meshing each sub-domain separately after the definition of the various domain interfaces and after a mesh of these interfaces has been constructed.

Provided with a fine mesh of the domain under interest, the posteriori partitioning method consists of splitting this mesh into several sub-meshes in order to distribute the computational effort over several processors, each of them being responsible

for the solution of a physical problem for one sub-domain. The global solution is achieved by merging all of the partial solutions. The most frequent case is element-based decomposition in which the fine mesh is partitioned by distributing the cells among the sub-domains i.e. one cell is logically associated with one and only one sub-domain. Another case is node-based decomposition in which the mesh is partitioned by distributing its nodes among the sub-domains, i.e. one node is logically associated with one and only one sub-domain. The main drawback of such a method is related to its memory requirement, as it is necessary to store the initial mesh and, at least, one of the sub-meshes. Nevertheless, the posteriori methods are widely used.

Of course, in practice, these parallelization approaches are often combined by taking into account their advantages and weaknesses.

These and different partition methods are presented in greater detail in the books of Frey and George (2008) and Lo (2015).

References

Azarenok, B. N. (2002). Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics. *SIAM Journal on Numerical Analysis*, *40*(40), 651–682.

Charakch'yan, A. A. (1993). Almost conservative difference schems for the equations of gas dynamics. *Computational Mathematics and Mathematical Physics*, *33*, 1473.

Charakch'yan, A. A. (1994). Compound difference schems for time-dependent equations on non-uniform nets. *Communications in Numerical Methods in Engineering*, *10*, 93.

Charakch'yan, A. A., & Ivanenko, S. A. (1988). A variational form of the Winslow grid generator. *Journal of Computational Physics*, *136*, 385–398.

Charakch'yan, A. A., & Ivanenko, S. A. (1997). A variational form of the Winslow grid generator. *Journal of Computational Physics*, *136*, 385–398.

Danaev, N. T., Liseikin, V. D., & Yanenko, N. N. (1980). Numerical solution on a moving curvilinear grid of viscous heat-conducting flow about a body of revolution. *Chisl. Metody Mekhan. Sploshnoi Sredy*, *11*(1), 51–61. (Russian).

Fletcher, C. A. J. (1997). *Computational Techniques for Fluid Dynamics 1: Fundamental and General Techniques*. Berlin: Springer.

Frey, P. J., & George, P.-L. (2008). *Mesh Generation Application to Finite Elements*. Verlag: ISTE Ltd and Wiley Inc.

Glasser, A. H., Liseikin, V. D., & Kitaeva, I. A. (2005). Control of grid properties with the help of monitor metrics. *Computational Mathematics and Mathematical Physics*, *45*(8), 1416–1432.

Glasser, A. H., Liseikin, V. D., Shokin, Ju. I., Vaseva, I. A., & Likhanova, Ju. V. (2006). *Grid Generation with the Use of Beltrami and Diffusion Equations*. Nauka, Novosibirsk. (Russian).

Glasser, A. H., & Tang, X. Z. (2004). The SEL macroscopic modeling code. *Computer Physics Communications*, *164*, 237–243.

Ivanenko, S. A. (1988). Generation of non-degenerate meshes. *USSR Computational Mathematics and Mathematical Physics*, *28*(5), 141.

Kovenya, V. M., Tarnavskii, G. A., & Chernyi, S. G. (1990). *Application of a Splitting Method to Fluid Problems*. Novosibirsk: Nauka. (in Russian).

Langtangen,. (2003). *Computational partial differential equations, numerical methods and diffpack programming*. Berlin: Springer.

Liseikin, V. D. (2014a). *Numerical Grids*. SD RAS, Novosibirsk (in Russia): Theory and Applications.

Liseikin, V. D. (2014b). *Methods for Generating Numerical Grids*. Novosibirsk: NSU. (in Russia).

Lo, S. H. (2015). *Finite Element Mesh Generation*. Boca Raton: CRC Press Taylor and Francis Group.

Lomonosov, I. V., Frolova, A. A., & Charakhch'yan, A. A. (1997). Computation of hight-velocity impact of thin foil upon conical target (survey). *A Mathematical Modeling*, *9*(5), 48–60. (Russian).

Vaseva, I. A., & Liseikin, V. D. (2011). Application of the finite element method for generating adaptive grids. *Computational Technologies*, *16*(5), 3–15. (in Russia).

Winslow, A. M. (1981): Adaptive mesh zoning by the equipotential method. UCID-19062, Lawrence Livermore National Laboratories.

Yanenko, N. N. (1971). *The method of fractional steps. The solution of problems of mathematical physics in several variables*. New York: Springer.