$$V = \left\{ v : v \text{ is a continuous function on } \Omega, \frac{\partial v}{\partial x_1} \text{ and } \frac{\partial v}{\partial x_2} \right.$$

$$\text{are piecewise continuous and bounded on } \Omega \,,$$

$$\left. \text{and } \int_\Omega v \, d\mathbf{x} = 0 \right\} .$$

To construct the finite element method for (1.25), let K_h be a triangulation of Ω as in the previous subsection. The finite element space V_h is defined by

$$V_h = \{ v : v \text{ is a continuous function on } \Omega \text{ and}$$
$$\text{is linear on each triangle } K \in K_h \} .$$

Note that the functions in V_h are not required to satisfy any boundary condition. Now, the finite element solution satisfies

$$\text{Find } p_h \in V_h \text{ such that } a(p_h, v) = (f, v) + (g, v)_\Gamma \qquad \forall v \in V_h \,. \qquad (1.28)$$

Again, for the pure Neumann boundary condition, V_h needs to be modified to

$$V_h = \left\{ v : v \text{ is a continuous function on } \Omega \text{ and is linear} \right.$$
$$\left. \text{on each triangle } K \in K_h, \text{ and } \int_\Omega v \, d\mathbf{x} = 0 \right\} .$$

As in the last two subsections, (1.28) can be formulated in matrix form, and an error estimate can be similarly stated under an appropriate smoothness assumption on the solution p that involves its second partial derivatives.

The *Poisson equation* has been considered in (1.16) and (1.25). More general partial differential equations will be treated in subsequent sections and chapters.

1.1.4 Programming Considerations

The essential features of a typical computer program implementing the finite element method are included in the following parts:

- Input of data such as the domain Ω, the right-hand side function f, the boundary data γ and g (cf. (1.25)), and the coefficients that may appear in a differential problem;
- Construction of the triangulation K_h;
- Computation and assembly of the stiffness matrix $\mathbf{A}$ and the right-hand side vector $\mathbf{f}$;
- Solution of the linear system of algebraic equations $\mathbf{Ap} = \mathbf{f}$;
- Output of the computational results.

The data input can be easily
result output depends on the
Here we briefly discuss the otl
two dimensions.

1.1.4.1 Construction of th

The triangulation K_h can be
initial coarse partition of Ω; fi
midpoints of edges of coarse
refinements will lead to *quasi-*
tially have the same size in al
of Ω is a curve, special care n

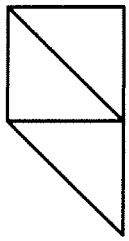

Fig. 1

In practical applications, i
vary considerably in size in di
smaller triangles in regions v
or where certain derivatives a
strategy is carried out. In this s
zone between regions with tria
local refinement results (i.e, r
an edge of another triangle; se
grids where needed are called
in Chap. 6.

Let a triangulation K_h hav
can be represented by two ar
1, 2) indicates the coordinates
($i = 1, 2, 3$) enumerates the n
example is demonstrated in Fi
in circles. For this example, t
$\mathcal{M} = 11$:

$$\mathcal{Z} = \begin{pmatrix} 1 & 1 & 2 \\ 2 & 4 & 5 \\ 4 & 3 & 4 \end{pmatrix}$$

The data input can be easily implemented in a small subroutine, and the result output depends on the computer system and software the user has. Here we briefly discuss the other three parts. As an illustration, we focus on two dimensions.

1.1.4.1 Construction of the Triangulation K_h

The triangulation K_h can be constructed from a successive refinement of an initial coarse partition of Ω; fine triangles can be obtained by connecting the midpoints of edges of coarse triangles, for example. A sequence of uniform refinements will lead to *quasi-uniform* grids where the triangles in K_h essentially have the same size in all regions of Ω (cf. Fig. 1.9). If the boundary Γ of Ω is a curve, special care needs to be taken of near Γ (cf. Sect. 1.5).

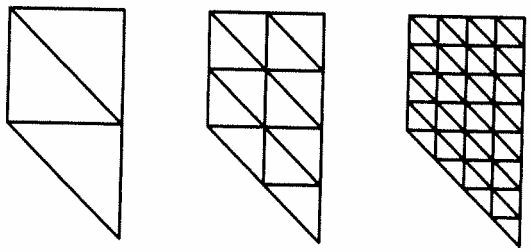

Fig. 1.9. Uniform refinement

In practical applications, it is often necessary to use triangles in K_h that vary considerably in size in different regions of Ω. For example, one utilizes smaller triangles in regions where the exact solution has a fast variation or where certain derivatives are large; see Fig. 1.10, where a *local refinement* strategy is carried out. In this strategy, proper care is taken of in the transition zone between regions with triangles of different sizes so that a so-called *regular* local refinement results (i.e. no vertex of one triangle lies in the interior of an edge of another triangle; see Chap. 6). Methods that automatically refine grids where needed are called *adaptive methods*, and will be studied in detail in Chap. 6.

Let a triangulation K_h have M nodes and $\mathcal{M}$ triangles. The triangulation can be represented by two arrays $\mathbf{Z}(2, M)$ and $\mathcal{Z}(3, \mathcal{M})$, where $\mathbf{Z}(i, j)$ $(i = 1, 2)$ indicates the coordinates of the jth node, $j = 1, 2, \ldots, M$, and $\mathcal{Z}(i, k)$ $(i = 1, 2, 3)$ enumerates the nodes of the kth triangle, $k = 1, 2, \ldots, \mathcal{M}$. An example is demonstrated in Fig. 1.11, where the triangle numbers are denoted in circles. For this example, the array $\mathcal{Z}(3, \mathcal{M})$ is of the form, where $M = \mathcal{M} = 11$:

$$\mathcal{Z} = \begin{pmatrix} 1 & 1 & 2 & 3 & 4 & 4 & 5 & 6 & 7 & 7 & 8 \\ 2 & 4 & 5 & 4 & 5 & 7 & 9 & 7 & 9 & 10 & 10 \\ 4 & 3 & 4 & 6 & 7 & 6 & 7 & 8 & 10 & 8 & 11 \end{pmatrix}.$$

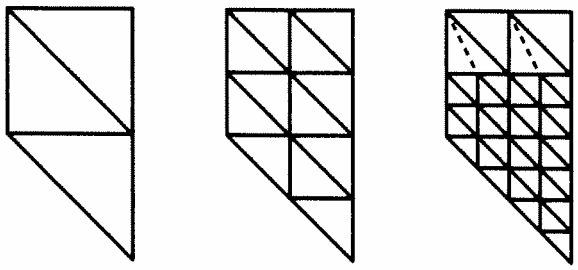

Fig. 1.10. Nonuniform refinement

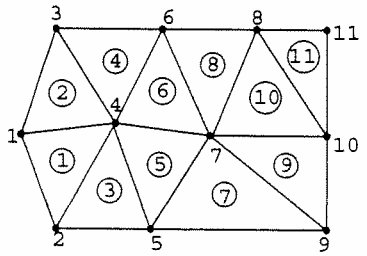

Fig. 1.11. Node and triangle enumeration

If a direct method (Gaussian elimination) is employed to solve the linear system $\mathbf{Ap} = \mathbf{f}$, the nodes should be enumerated in such a way that the *band width* of each row in $\mathbf{A}$ is as small as possible. This matter will be studied in Sect. 1.10, in connection with the discussion of solution methods for linear systems.

In general, when local refinement is involved in a triangulation K_h, it is very difficult to enumerate the nodes and triangles efficiently; some strategies will be given in Chap. 6. For a simple domain Ω (e.g., a convex polygonal Ω), it is rather easy to construct and represent a triangulation that utilizes uniform refinement in the whole domain.

1.1.4.2 Assembly of the Stiffness Matrix

After the triangulation K_h is constructed, one computes the *element stiffness matrices* with entries a_{ij}^K given by (1.23). We recall that $a_{ij}^K = 0$ unless nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ are both vertices of $K \in K_h$.

For a kth triangle K_k, $\mathcal{Z}(m,k)$ $(m = 1,2,3)$ are the numbers of the vertices of K_k, and the element stiffness matrix $\mathbf{A}^{(k)} = \left(a_{mn}^k\right)_{m,n=1}^3$ is now calculated as follows:

$$a_{mn}^k = \int_{K_k} \nabla\varphi_m \cdot \nabla\varphi_n \, d\mathbf{x}, \quad m,\ n = 1,2,3\,,$$

where the (linear) basis function φ_m on K_k satisfies

$\varphi_m(\mathbf{x}_{\mathcal{Z}($

The right-hand side on K_k is

$$f_m^k = \int_K$$

Note that m and n are the loc
and j used in (1.23) are the g

To assemble the global mat
one loops over all triangles K_l
different $K_k's$:

For $k = 1, 2,$

$a_{\mathcal{Z}(m,k),\mathcal{Z}(n,k}$

$f_{\mathcal{Z}(m,k)} = f_{\mathcal{Z}}$

The approach used is *element-*
angles). Experience shows tha
oriented approach (i.e., loopir
too much time in repeated co

1.1.4.3 Solution of a Linea

The solution of the linear sy
method (Gaussian eliminatio
gradient method), which will b
that in use of these two met
$\mathbf{A}(M,M)$ to store the stiffne
usually a band matrix, only t
in an one-dimensional array.

1.2 Sobolev Spaces

In the previous section, an intr
for two simple model problem
general formulation, we need t
the development of the functio
spaces of continuous functions
in the previous section. We est
sufficient to develop the found
in this book.

$$\varphi_m(\mathbf{x}_{\mathcal{Z}(n,k)}) = \begin{cases} 1 & \text{if } m = n\,, \\ 0 & \text{if } m \neq n\,. \end{cases}$$

The right-hand side on K_k is computed by

$$f_m^k = \int_{K_k} f\varphi_m \, d\mathbf{x}, \quad m = 1, 2, 3\,.$$

Note that m and n are the local numbers of the three vertices of K_k, while i and j used in (1.23) are the global numbers of vertices in K_h.

To assemble the global matrix $\mathbf{A} = (a_{ij})$ and the right-hand side $\mathbf{f} = (f_j)$, one loops over all triangles K_k and successively adds the contributions from different $K_k's$:

For $k = 1, 2, \ldots, \mathcal{M}$, compute

$$a_{\mathcal{Z}(m,k),\mathcal{Z}(n,k)} = a_{\mathcal{Z}(m,k),\mathcal{Z}(n,k)} + a_{mn}^k\,,$$
$$f_{\mathcal{Z}(m,k)} = f_{\mathcal{Z}(m,k)} + f_m^k, \quad m,\ n = 1, 2, 3\,.$$

The approach used is *element-oriented*; that is, we loop over elements (i.e., triangles). Experience shows that this approach is more efficient than the *node-oriented* approach (i.e., looping over all nodes); the latter approach wastes too much time in repeated computations of $\mathbf{A}$ and $\mathbf{f}$.

1.1.4.3 Solution of a Linear System

The solution of the linear system $\mathbf{Ap} = \mathbf{f}$ can be performed via a direct method (Gaussian elimination) or an iterative method (e.g., the conjugate gradient method), which will be discussed in Sect. 1.10. Here we just mention that in use of these two methods, it is not necessary to exploit an array $\mathbf{A}(M, M)$ to store the stiffness matrix $\mathbf{A}$. Instead, since $\mathbf{A}$ is sparse and usually a band matrix, only the nonzero entries of $\mathbf{A}$ need to be stored, say, in an one-dimensional array.

1.2 Sobolev Spaces

In the previous section, an introductory finite element method was developed for two simple model problems. To present the finite element method in a general formulation, we need to use function spaces. This section is devoted to the development of the function spaces that are slightly more general than the spaces of continuous functions with piecewise continuous derivatives utilized in the previous section. We establish the small fraction of these spaces that is sufficient to develop the foundation of the finite element method as studied in this book.