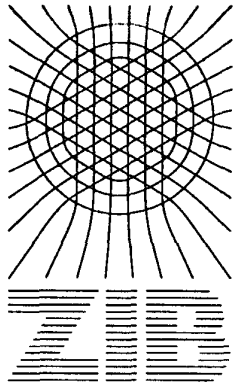
Konrad-Zuse-Zentrum für Informationstechnik Berlin



Ralf Kornhuber Rainer Roitzsch

On Adaptive Grid Refinement in the Presence of Internal or Boundary Layers

Preprint SC 89-5 (November 1989)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Strasse 10
1000 Berlin 31
Verantwortlich: Dr. Klaus André
Umschlagsatz und Druck: Rabe KG Buch- und Offsetdruck Berlin

ISSN 0933-7911

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10
1000 Berlin 31
Verantwortlich: Dr. Klaus André
Umschlagsatz und Druck: Rabe KG Buch-und Offsetdruck Berlin

ISSN 0933-7911

Ralf Kornhuber Rainer Roitzsch

On Adaptive Grid Refinement in the Presence of Internal or Boundary Layers

Abstract

We propose an anisotropic refinement strategy which is specially designed for the efficient numerical resolution of internal and boundary layers. This strategy is based on the directed refinement of single triangles together with adaptive multilevel grid orientation. Compared to usual methods, the new anisotropic refinement ends up in more stable and more accurate solutions at much less computational cost. This is demonstrated by several numerical examples.

Keywords: Adaptive finite elements, directed refinement, adaptive grid orientation, convection diffusion equation, internal and boundary layers.

Subject Classification: AMS(MOS): 35J70, 35L67, 65N30, 76D30

Contents

Introduction

Finite element methods for linear elliptic boundary value problems fix an approximate solution in some finite dimensional trial space, requiring that the residual is orthogonal to a suitable subspace of test functions. Adaptive methods appropriately enlarge the trial space if the obtained approximation is deemed too inaccurate.

For simplicity, we will choose the approximate solutions to be continuous and piecewise linear on some triangulation of the underlying polygonal domain. Then the enlargement of the trial space is equivalent to the refinement of some suitably selected triangles. In this way starting from some initial triangulation, a certain refinement strategy produces a sequence of triangulations which are hoped to suit better and better to the given problem.

As the refinement of single triangles determines the local structure of the resulting triangulation, it should reflect as much as possible the expected local behaviour of the exact solution. In this sense the well–known regular (red) refinement introduced by Bank [5] is well matched with the local character of isotropic or almost isotropic problems.

On the other hand many practical problems in the field of fluid dynamics, combustion, or semi–conductor device simulation turn out to be anisotropic as the principal elliptic part is dominated by lower order convective terms. As a consequence internal and boundary layers are typically occurring in the solutions. In numerical approximations accuracy implies a very small mesh size only in a certain refinement direction orthogonal to the actual layer. Following the layer the mesh size may be comparably large. Hence the adaptive generation of optimal triangulations can be achieved only by special anisotropic refinement strategies, as indicated by Hackbusch [11], p. 226. Nevertheless convection dominated problems are frequently treated with standard isotropic refinements, leading to lots of problems both in complexity and stability.

On this background the present paper is concerned with the further investigation of a suitable anisotropic refinement strategy that has been recently introduced by the authors in [15]. After a short description of the usual isotropic refinement, the so–called directed (blue) refinement is introduced in the second chapter. This refinement may be regarded as a generalization of the bisection of only one of the two step sizes in case of a rectangular grid. As blue refinement should be performed only close to the numerical layer given by the approximate solution, the numerical layer has to be located somehow in the triangulation. A straightforward algorithm to construct a so–called discrete layer consisting of edges of the triangulation is given in Section 3.1.

To make sure that blue refinement is feasibled and is carried out in the desired direction, the edges constituting the discrete layer have to be adjusted to the numerical layer which is usually not known a priori. A corresponding adaptive grid orientation is described in Section 3.2. The desired anisotropic refinement strategy is finally compiled in the fourth chapter. Then Chapter 5 contains some remarks on implementation and data structures. In the final chapter we present some numerical examples demonstrating the efficiency of the proposed method.

The authors want to express their sincere thanks to S. Wacker for her careful typing of the manuscript and P. Deuflhard and our colleagues at the Konrad–Zuse–Zentrum for lots of stimulating discussions.

1. Isotropic refinement

Let Ω be a polygonal domain in $\mathbb{R}^2$. Then a triangulation $\mathcal{T}$ of Ω is a set of triangles such that Ω is the union of all triangles $t \in \mathcal{T}$ and such that the intersection of two triangles t, $t' \in \mathcal{T}$ either consists of a common edge or a common vertex or is empty. The set of all continuos functions on Ω which are linear on each $t \in \mathcal{T}$ is called $\mathcal{S}(\mathcal{T})$. To obtain a sufficiently accurate solution we intend to produce a sequence of triangulations $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, ... by successive refinement of a given initial triangulation $\mathcal{T}_0$. Thus the triangulation $\mathcal{T}_{k+1}$ on (refinement) level $k+1$ results from the application of a certain refinement strategy to $\mathcal{T}_k$. A refinement strategy consists of some manipulation of an actual triangulation including the refinement of single triangles, together with the detection of suitable a posteriori information which is connecting the actual refinement to the given problem. In the sequel we will be mainly interested in the question how to manipulate a triangulation, pointing out what a posteriori information is used in the different steps.

Let us roughly recall a well-known refinement strategy due to Bank et al. [4], referring to Bank [2], Leinen[16], and Deuflhard et al.[9] for recent developments.

The presented strategy inherits its isotropic character from the regular (red) refinement which is performed by dividing a triangle t into four similar subtriangles, as shown in Figure 1.1. These subtriangles will be called sons of the triangle t.

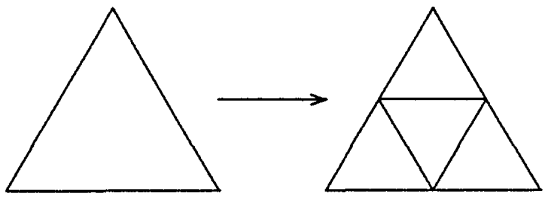


Figure 1.1 Red refinement

Note that in case of a rectangular grid the bisection of the stepsizes h_x, h_y is recovered. To remedy irregular nodes an additional irregular (green) refinement is required which is illustrated in Figure 1.2 .

Now the isotropic refinement of a triangulation $\mathcal{T}_k$ on level k reads as follows.

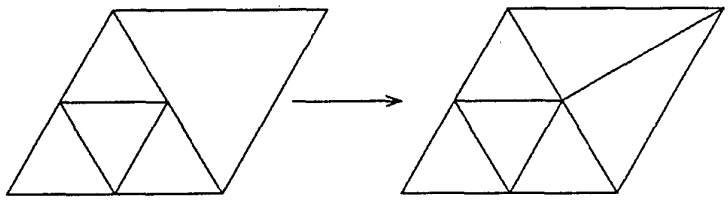


Figure 1.2 Green refinement

Algorithm 1.1: Isotropic refinement

Step 1: If $k > 0$, undo all green refinements.

Step 2: Mark a subset of $\mathcal{T}_k$ for refinement.

Step 3: Apply the red refinement to all marked triangles.

Step 4: Continue the red refinement of such triangles $t \in \mathcal{T}$ which possess more than one refined neighbour.

Step 5: Apply the green refinement to all triangles with one refined neighbour.

Obviously the first step guarantees a minimal angle condition. Suitable a posteriori error estimates should be incorporated in Step 2.

As indicated in [11] this construction does not necessaryly produce a nested sequence of triangulations. But following Bank et al. [3], a possibly different, nested sequence $\mathcal{T}_0 = \mathcal{T}_0', \mathcal{T}_1', \ldots, \mathcal{T}_{k-1}', \mathcal{T}_k' = \mathcal{T}_k$, being uniquely determined by $\mathcal{T}_0$ and $\mathcal{T}_k$, may be constructed a posteriori.

2. Directed refinement

In this chapter we will introduce a refinement of single triangles, giving preference to a certain direction which has to be derived in advance from a posteriori information.

Let us first consider the most simple case of a rectangular grid. Clearly the uniform refinement in direction of the y–axis is done by bisecting only h_y. Applied to the corresponding triangulation, this leads to the directed or blue refinement which is illustrated in Figure 2.1. .

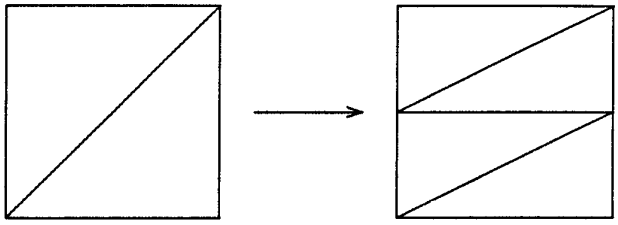


Figure 2.1 Blue refinement of a rectangular grid

Obviously the triangles resulting from directed refinement are no longer a subdivision of a single triangle, but of the quadrangle formed by two triangles with a common diagonal edge. Thus directed refinement always works on a couple of two triangles, making the situation more complicated than in the regular case.

A formal generalization of blue refinement to any two triangles with a common edge does not make much sense, and is not even feasible. To incorporate a direction in the local refinement we first state the following definition.

Definition 2.1 Let a given direction g be orthogonal to the edge e of the triangle t. If only the bisection of at least one of the two remaining edges is intended, we say that t is marked for refinement in direction g.

Note that Definition 2.1 requires some previous selection of edges for refinement. Now we are ready to extend the blue refinement to a more general situation.

Definition 2.2 Let t, t' be triangles with a common edge called diagonal and associated directions $g = g(t)$ and $g' = g(t')$. Suppose that the following conditions hold.

(B1) t and t' are marked for refinement in the directions g and g'.

(B2) Both triangles t and t' have a common refinement direction $g = g'$ which is not orthogonal to the diagonal.

(B3) If the diagonal is removed, the resulting quadrangle does not degenerate to a triangle.

Then the directed (blue) refinement of t, t' is defined according to Figure 2.2.

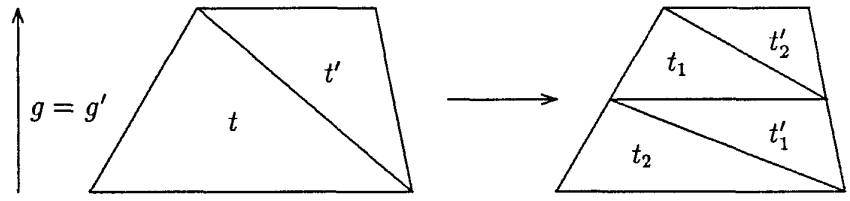


Figure 2.2 Blue refinement

The triangles t_1, t_2 and t'_1, t'_2 are called sons of t and t', respectively.

Note that (B1) refers to a posteriori information, according to Definition 2.1. The other conditions (B2) and (B3) make sure that the triangles t and t' are well matched. In particular blue refinement of t, t' is defined only if a trapezium is formed by the union of t and t'.

Remark 2.3 According to Definition 2.1 the condition (B1) implies that the direction of the edges is connected with the refinement direction. As the latter is depending on the approximate solution, the triangulation may have to be rearranged adaptively. We will come back to this problem in the next chapter.

Remark 2.4 Obviously the equality of directions required in (B1) and (B2) will never hold in numerical practise. Hence in the numerical experiments reported below the conditions (B1) and (B2) are weakened in the sense that an angle φ is regarded as zero if $|\sin(\varphi)| < \theta$ with $\theta = 0.15$.

Remark 2.5 If blue refinement is involved in the generation of a sequence of triangulations $\mathcal{T}_0, \mathcal{T}_1 \ldots, \mathcal{T}_k$, $k > 0$, then it is not possible to reconstruct a nested sequence of triangulations $\mathcal{T}_0 = \mathcal{T}'_0, \mathcal{T}'_1, \ldots, \mathcal{T}'_{k-1}, \mathcal{T}'_k = \mathcal{T}_k$ as in the isotropic case.

We now turn to the investigation of the interior angles resulting from successive blue refinement. Here we will use the notations introduced in Figure 2.3.

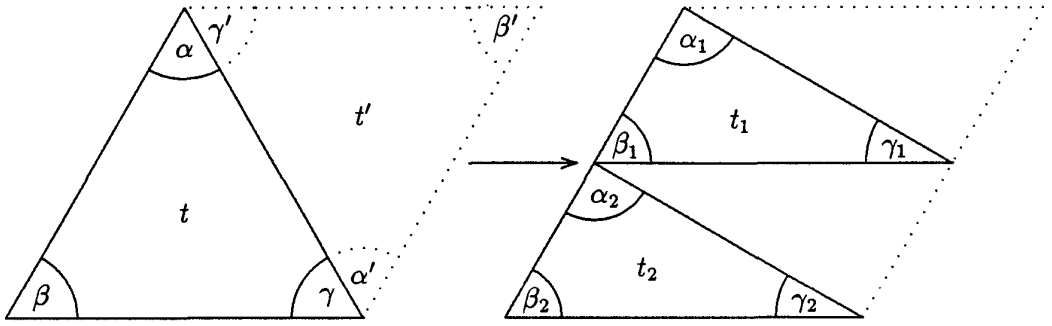


Figure 2.3 Evolution of interior angles

Lemma 2.6 Suppose that blue refinement of t, t' in vertical direction g is possible according to (B1), (B2), (B3) and that the supplementary condition (B4) is fulfilled.

(B4) The angles in t, t' satisfy the relations $\beta \geq \min(\alpha, \gamma)$ and $\beta' \geq \min(\alpha', \gamma')$.

Then the estimate

$$\max(\alpha_i, \beta_i, \gamma_i) + \min(\beta_i, \gamma_i) \leq \max(\alpha, \beta, \gamma) + \min(\beta, \gamma) \qquad (2.1)$$

holds for $i = 1, 2$.

Proof: Obviously we have

$$\beta_i = \beta\ , \quad \alpha_i + \gamma_i = \alpha + \gamma\ , \quad i = 1, 2 \qquad (2.2)$$

Now with the help of (2.2), the assertion is checked for each relation of the sizes of α, β, γ that is allowed by (B4).
For example consider the case $\alpha \leq \beta \leq \gamma$. Then

$$\max(\alpha, \beta, \gamma) + \min(\beta, \gamma) = \alpha_i + \beta_i + \gamma_i - \alpha\ , \quad i = 1, 2. \qquad (2.3)$$

Let $\gamma_i = \max(\alpha_i, \beta_i, \gamma_i)$. Then $\beta_i = \min(\beta_i, \gamma_i)$, and the assertion follows from $\alpha_i \geq \alpha$. The remaining cases are treated in a similar way. ∎

As a direct consequence of Lemma 2.6 we obtain the following maximal angle condition. Here successive refinement of a triangle t is understood as subsequent refinement of t and its actual descendants.

Proposition 2.7 Let $t^{(0)}, t^{(1)}, t^{(2)}, \ldots$ denote a sequence of triangles resulting from successive blue refinement of $t = t^{(0)}$ in the same direction g according to the conditions (B1), ..., (B4). Then

$$\max(\alpha^{(j)}, \beta^{(j)}, \gamma^{(j)}) \leq \max(\alpha, \beta, \gamma) + \min(\beta, \gamma) \ , \quad j = 0, 1, \ldots \tag{2.4}$$

holds, with $\alpha^{(j)}, \beta^{(j)}, \gamma^{(j)}$ denoting the interior angles of $t^{(j)}$, $j = 0, 1, \ldots$.

Proof: The proof is obvious from (2.1). ∎

Remark 2.8 The estimate (2.4) is sharp, if $\alpha = \max(\alpha, \beta, \gamma)$, but may be much too pessimistic if $\alpha < \max(\alpha, \beta, \gamma)$. In the latter case it can be shown by elementary considerations that

$$\alpha_i + \min(\beta_i, \gamma_i) \leq \tfrac{3}{4}\pi \ , \quad i = 1, 2 \ , \tag{2.5}$$

with the denotations taken from Figure 2.3. Now it is straightforward to prove the following extension of the estimate (2.4). Under the assumptions of Proposition 2.7 we have

$$\max(\alpha^{(j)}, \beta^{(j)}, \gamma^{(j)}) \leq \max\{\max(\alpha, \beta, \gamma) + \min(\alpha, \beta, \gamma), \tfrac{3}{4}\pi\} \ , \tag{2.6}$$

for $j = 0, 1, \ldots$.

We have seen that interior angles remain bounded away from π under successive blue refinement. On the other hand blue refinement may lead to arbitrary stretched triangles so that the green closure may produce arbitrary obtuse angles as illustrated in Figure 2.4.

Figure 2.4 Green refinement in case of acute angles

For this reason green refinement of a triangle t is allowed only if the following condition is satisfied.

(G) t is resulting from an initial triangle by not more than q blue refinements.

Here the parameter q is indicating the degeneracy to be expected. We have chosen $q = 1$ for our numerical examples.

Remark 2.9 As a consequence of condition (G) all triangles with only one refined edge and more than q blue predecessors are refined in the regular way. Note that this strategy may bring about avalanches of forced red refinement on higher levels. Indeed every red refinement caused by (G) may lead again to the forced refinement of neighbouring triangles. This problem does not occur, if irregular nodes are introduced which is common practice in the case of quadrilateral elements.

Let us complete this chapter by some remarks on stretched finite elements as created by successive blue refinements. As we have seen above, interior angles remain bounded away from π by a constant depending on the initial triangulation. Hence a maximum angle condition is always fulfilled. But still the resulting angles may become arbitrary small, violating the minimum angle condition that is usually required in finite elements.

Now it is well–known that the order of the interpolation error is not affected by small angles (see [20], [13] or [1]). Only a maximum angle condition is required.

On the other hand almost degenerate elements are commonly suspected to increase the condition number of the corresponding stiffness matrix (see for instance [21] p. 17). We will give a simple example, showing that the condition number might even be improved, if the triangles are degenerating in an appropriate way.

Let us first consider the discretization of the Laplacian

$$\begin{aligned} u_{xx} + u_{yy} &= 1 \quad && \text{on } \Omega = [0,1] \times [0,1], \\ u|_\Gamma &= 0 && \text{on } \Gamma = \partial\Omega \end{aligned} \tag{2.7}$$

by linear finite elements on a uniform rectangular grid, generating the standard five–point difference scheme. Now both the condition number, behaving like $\mathcal{O}(1/h_x^2 + 1/h_y^2)$, and the L^2–error of order $\mathcal{O}(h_x^2 + h_y^2)$ are minimized for equal stepsizes $h_x = h_y$, or equivalently for nice triangulations.

Now the same argument leads to very different results in the case of anisotropic problems which are of special interest here. Replacing problem (2.7) by

$$\begin{aligned} \varepsilon u_{xx} + u_{yy} &= 1 \quad && \text{on } \Omega = [0,1] \times [0,1], \\ u|_\Gamma &= 0 && \text{on } \Gamma = \partial\Omega \end{aligned} \tag{2.8}$$

with $0 < \varepsilon \ll 1$ we obtain a condition number of order $\mathcal{O}(\varepsilon/h_x^2 + 1/h_y^2)$ and an L^2–error bound of $C(\varepsilon)(h_x^2/\varepsilon + h_y^2)$ by simple transformation of variables. Hence in this case the choice of $h_x = \sqrt{\varepsilon} h_y$ turns out to be optimal, resulting in degenerating elements for $\varepsilon \to 0$.

A straightforward extensionof the above reasoning covers complicated situations. Assuming as in (B1) that the triangulation is locally oriented to the

actual layer, we obtain that the size of neighbouring angles should be of order of the width of the layer.

Finally these heuristic arguments are strengthened by numerical experiments showing that the anisotropic refinement strategy involving blue refinements ends up in much better conditioned systems than the usual isotropic technique.

3. Local grid orientation

Blue refinements, as introduced in the preceding chapter, are intended to render a both efficient and highly accurate resolution of an interior or boundary layer. Hence blue refinements should be applied only in the neighbourhood of this layer, while the usual red refinements are appropriate in the remaining parts of the domain. For this reason a suitable neighbourhood of the layer has to be detected from the actual approximate solution. In addition blue refinement is allowed by (B1) only if the actual triangulation is locally oriented in the direction of the layer.

To meet these difficulties, actual blue refinements have to be prepared by the following two steps. First the numerical layer is approximated by a polygonal consisting of edges of the actual triangulation which is called discrete layer. In the next step the discrete layer is further adjusted by a multilevel grid orientation. Finally all triangles neighbouring the oriented discrete layer are candidates for blue refinement. A detailed description of the whole procedure is given in the following two sections.

3.1 Construction of a discrete layer

Let $u : \Omega \to \mathbb{R}$ be some real valued function on Ω. Then a curve $\gamma \subset \Omega$ with the property that $\| \operatorname{grad} u \|$ is descending much quicker in both normal directions than in tangential direction of γ is called layer of u. In the sequel we assume that u is the exact solution of our given problem with a single layer $\gamma \subset \Omega$ which allows for a bijective parametrization. Hence bifurcations and loops are excluded so that no interaction of layers is considered.

Assume that a hierarchy of triangulations $\mathcal{T}_0, \ldots \mathcal{T}_k$, $k \geq 1$ has been constructed and that an approximate solution U_k has been computed on $\mathcal{T}_k$. We prepare the construction of the discrete layer by some notations.

Let $\mathcal{E}_k$ and $\mathcal{P}_k$ denote the sets of edges and vertices of triangles in $\mathcal{T}_k$. We will write $t = [e_0, e_1, e_2]$ or $e = [p_0, p_1]$, representing a triangle $t \in \mathcal{T}_k$ by its edges $e_0, e_1, e_2 \in \mathcal{E}_k$ or an edge $e \in \mathcal{E}_k$ by its ends $p_0, p_1 \in \mathcal{P}_k$, respectively. This notation does not express a certain orientation. If a node $p \in \mathcal{P}_k$ is resulting from the bisection of an edge $E \in \mathcal{E}_{k-1}$, we write $p = m(E)$, calling p the midpoint of E.

The sets $\mathcal{T}_k$, $\mathcal{E}_k$, $\mathcal{P}_k$ can be decomposed in the disjunct subsets $\mathcal{T}_k^j$, $\mathcal{E}_k^j$, $\mathcal{P}_k^j$, $j = 0, \ldots, k$, where

$$\begin{aligned} \mathcal{T}_k^0 &= \mathcal{T}_k \cap \mathcal{T}_0 \\ \mathcal{T}_k^j &= \{t \in \mathcal{T}_k \cap \mathcal{T}_j \wedge t \notin \mathcal{T}_{j-1}\} \quad j = 1, \ldots, k \end{aligned} \tag{3.1}$$

and $\mathcal{E}_k^j$, $\mathcal{P}_k^j$, $j = 0, \ldots, k$, are defined analogously. Following Bank et al.[3]

the elements of $\mathcal{T}_k^j$, $\mathcal{E}_k^j$ and $\mathcal{P}_k^j$ are called level j triangles, edges and vertices, respectively.

A triangle t is said to have (refinement) depth j, if it is resulting from j successive refinements of an initial triangle $T \in \mathcal{T}_0$. The subset of all triangles $t \in \mathcal{T}_k$ with maximal depth is called $\mathcal{T}_{k,k}$. Finally $\mathcal{P}_{k,k}$ and $\mathcal{E}_{k,k}$ denote the subsets of all vertices and edges of triangles $t \in \mathcal{T}_{k,k}$, respectively.

Now we are ready to give a precise definition of a discrete layer.

Definition 3.1 A polygonal approximation γ_k of a layer γ consisting of edges $e_i = [p_{i-1}, p_i] \in \mathcal{E}_k^* := \mathcal{E}_k^k \cap \mathcal{E}_{k,k}, i = 1, \ldots, n$, with vertices $p_i \in \mathcal{P}_k^* := \mathcal{P}_k^k \cap \mathcal{P}_{k,k}, i = 0, \ldots, n$, is called discrete layer. We write $\gamma_k = (e_i)_{i=1}^n$ or $\gamma_k = (p_i)_{i=0}^n$, equivalently.

Note that the approximation of γ is performed only in the region of maximal depth.

Remark 3.2 Each $p \in \mathcal{P}_k^k$ originates from the subdivision of an edge $E \in \mathcal{E}_{k-1}$. We define the subset $\mathcal{E}_{k-1}^{(*)} \subset \mathcal{E}_{k-1}$ by

$$\mathcal{E}_{k-1}^{(*)} = \{E \in \mathcal{E}_{k-1} | m(E) \in P_k^*\} . \tag{3.2}$$

Then we may equivalently represent the discrete layer $\gamma_k = (p_i)_{i=0}^n$ by $\gamma_k = (E_i)_{i=0}^n$ with $E_i \in \mathcal{E}_{k-1}^{(*)}$ and $p_i = m(E_i)$, $i = 0, \ldots, n$. This representation is illustrated in Figure 3.1 and will be frequently used in the sequel.

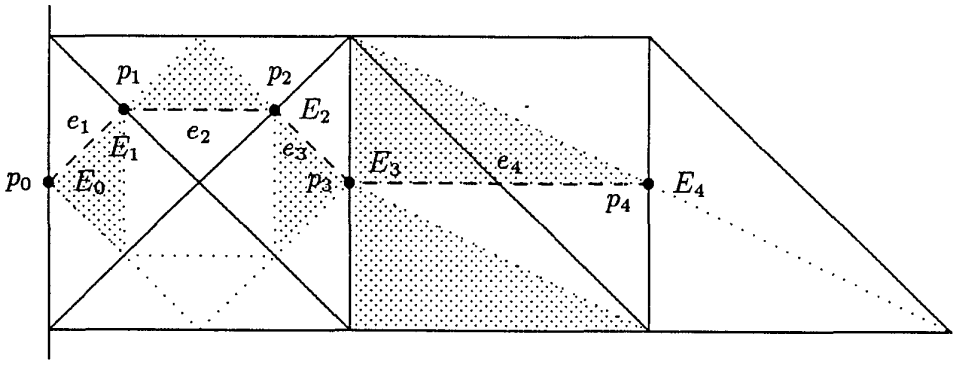


Figure 3.1 Discrete layer

A straightforward way to establish a discrete layer γ_k consists of the following two steps.

- Determine a setup point $p^{(0)} \in \mathcal{P}_k^*$ as close as possible to the numerical layer detected from the approximate solution U_k.

- Starting from $p^{(0)}$, follow the numerical layer in its two directions to fix the remaining vertices $p^{(\pm i)}$, $i = 1, \ldots, m^{\pm}$.

Finally we may rename the vertices $p^{(i)}, i = -m^-, \ldots, m^+$, to obtain $\gamma_k = (p_i)_{i=0}^n$. Let us now study how to perform each of these steps.
As the detection of a posteriori information is not our primary interest here, we simply characterize the discrete layer by the occurrence of steep gradients. The main idea is to represent γ_k as a sequence of edges $E_i \in \mathcal{E}_{k-1}^{(*)}$, $i = 0, \ldots, n$ which are intersecting the numerical layer. Hence $\mu(E_i), i = 0, \ldots, n$, with μ defined by

$$\mu(E) = \frac{|U_k(P_1) - U_k(P_2)|}{\|P_1 - P_2\|}, \quad E = [P_1, P_2] \in \mathcal{E}_{k-1}^{(*)} \tag{3.3}$$

should be large. This is leading to the following algorithm to construct $\gamma_k = (E_i)_{i=0}^n$.

Algorithm 3.3: Construction of γ_k

Step 1: Determine a setup edge $E^{(0)} \in \mathcal{E}_{k-1}^{(*)}$ with the property

$$\mu(E^{(0)}) = \max_{E \in \mathcal{E}_{k-1}^{(*)}} \mu(E)\,, \tag{3.4}$$

together with a setup triangle $T^{(0)} \in \mathcal{T}_{k-1}$ neighbouring $E^{(0)}$.

Step 2: If $T^{(0)} = [E^{(0)}, E_1, E_2]$ is blue refined, then $E^{(1)}$ and $T^{(1)}$ are determined according to Fig 3.2. Else select $E^{(1)}$, such that

$$\mu(E^{(1)}) = \max\{\mu(E_1), \mu(E_2)\} \tag{3.5}$$

holds. Then $T^{(1)}$ is the triangle neighbouring $T^{(0)}$ and $E^{(1)}$.

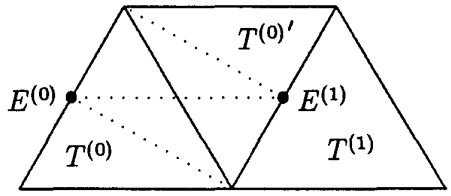


Figure 3.2 Construction of γ_k

Step 3: Repeat Step 2 inductively with the index 0 replaced by $i = 1, 2, \ldots$, until $T^{(i+1)}$ is not refined or the boundary of Ω is reached. In this case, set $m^+ = i$ and continue with the next step.

Step 4: Denote the triangle neighbouring $T^{(0)}$ and $E^{(0)}$ by $T^{(-1)}$. Then repeat the process described in Step 2 and Step 3 with $T^{(0)}$ replaced by $T^{(-1)}$ to construct the sequence $E^{(i)}, i = -1, \ldots, m^-$.

We say that a triangle t is neighbouring another, if both have a common edge and t is neighbouring an edge e, if $t = [e, e_1, e_2]$.

3.2 Multilevel grid orientation

Suppose that a discrete layer $\gamma_k = (p_i)_{i=0}^n \subset \mathcal{P}_k^*$ can be constructed in some suitable way. Recall that γ_k may equivalently be written as $\gamma_k = (E_i)_{i=0}^n \subset \mathcal{E}_{k-1}^{(*)}$ or $\gamma_k = (e_i)_{i=1}^n \subset \mathcal{E}_k^*$. In the sequel, we assume that γ_k has no loops or equivalently that $p_i \neq p_j$ for $i \neq j$ and $i, j = 0, \ldots, n$. Further suppose that a refinement direction $g(t)$ is given for every $t \in \mathcal{T}_{k,k}$. It follows from the definition of the layer γ, that the level curves of u are a good approximation of $g_\perp$. Hence the choice

$$g(t) = \operatorname{grad} U_k|_t, t \in \mathcal{T}_{k,k}, \tag{3.6}$$

will be used in the sequel.

As $\gamma_k = (e_i)_{i=1}^n$ is constructed to approximate the layer γ, it is natural to start the search for pairs of triangles for possible blue refinement with the triangles t_i and t^i, neighbouring the edges e_i, $i = 1, \ldots, n$. Now recall that according to Definition 1.2 and (B1) the condition

$$(e_i, g(t)) = 0, \quad t = t_i, t^i \tag{3.7}$$

is necessary for the blue refinement of t_i or t^i with $(\cdot, \cdot)$ denoting the usual scalar product in $\mathbb{R}^2$. For this reason, either the layer γ must be known a priori, so that a corresponding choice of the initial triangulation $\mathcal{T}_0$ is possible, or the directions of $e_i, i = 1, \ldots, n$ have to be adjusted to $g(t), t = t_i, t^i$ in advance of an actual refinement step. For example, in case of a uniform, vertical refinement direction the part of a triangulation shown in Figure 3.1 should be transformed in the oriented triangulation illustrated in Figure 3.3. This gives rise to a multilevel grid orientation that will be described in the sequel.

Let us start with the two level orientation of a triangulation $\mathcal{T}_k$. Assume that an approximate solution U_k has been computed on $\mathcal{T}_k$ with $k > 0$. Then an oriented triangulation $\mathcal{T}_k^*$ is obtained as follows.

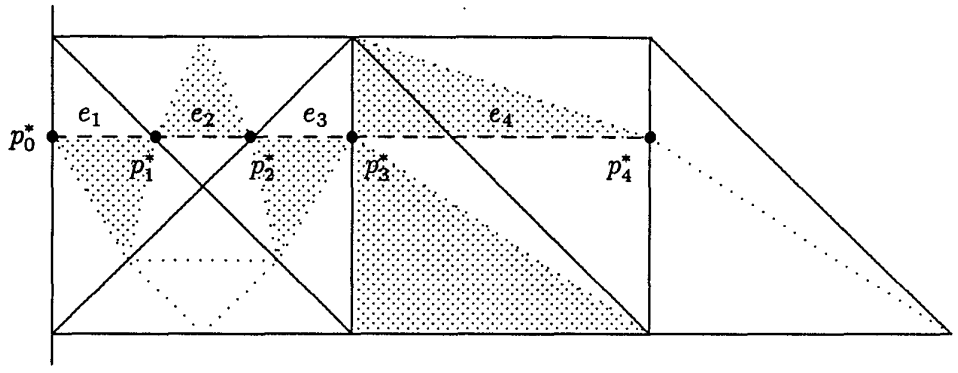


Figure 3.3 Adjusted discrete layer

Algorithm 3.4: Two level grid orientation

Step 1: Construct a discrete layer $\gamma_k = (p_i)_{i=0}^n \subset \mathcal{P}_k^*$ which is equivalently written as $\gamma_k = (E_i)_{i=0}^n \subset \mathcal{E}_{k-1}^{(*)}$ or $\gamma_k = (e_i)_{i=1}^n \subset \mathcal{E}_k^*$.

Step 2: Determine a sequence of directions G_i, $i = 1, \ldots, n$, by suitable restriction of the refinement directions $g(t), t \in \mathcal{T}_{k,k}$ to the edges e_i, $i = 1, \ldots, n$.

Step 3: If possible, determine a sequence of vertices $p_i^*, i = 0, \ldots, n$, with the following properties

$$p_i^* \in E_i, \; i = 0, \ldots, n \tag{3.8}$$

$$(e_i^*, G_i) = 0, \quad e_i^* = [p_{i-1}^*, p_i^*], \; i = 1, \ldots, n \tag{3.9}$$

and

$$\sum_{i=0}^{n} \|p_i^* - p_i\|^2 = \min, \tag{3.10}$$

where the minimum in (3.10) is taken over all sequences of vertices satisfying (3.8) and (3.9). Then $\mathcal{T}_k^*$ is obtained by moving the vertices p_i to $p_i^*, i = 0, \ldots, n$.

Else the two grid orientation failed and $\mathcal{T}_k$ is left unchanged.

Step 4: Compute $U_k^* \in \mathcal{S}(\mathcal{T}_k^*)$ by interpolation of $U_k \in \mathcal{S}(\mathcal{T}_k)$.

Remark 3.5 Clearly condition (3.7) results from (3.9) only if $G_i = g(t)$, $t = t_i, t^i$. Even if exact equality is weakened for numerical purposes, it follows that blue refinement will be performed only if the direction of the numerical layer varies very little in the neighbourhood of γ_k..

Let us consider each step of Algorithm 3.4 in some detail.

The construction of a discrete layer has been discussed in the preceding section.

Let us discuss the choice of a suitable restriction of the refinement direction g. As g is constant on each triangle $t \in \mathcal{T}_k$ a simple restriction of order 0 is well suited in this case (see for example Wesseling [23]). Note that each $e_i \in \gamma_k$ is intersecting either a red refined triangle $T_i \in \mathcal{T}_{k-1}$ or a quadrangle $Q_i = T_i \cup T_i'$ of a pair of blue refined triangles T_i, $T_i' \in \mathcal{T}_{k-1}$. Let t_i^j, $j = 1, \ldots, 4$, denote the sons of T_i or T_i, T_i', respectively. Then G_i is defined by

$$G_i = G_i(P) = \sum_{j=1}^{4} \frac{\alpha(t_i^j)}{\alpha(P)} g(t_i^j), \quad P = T_i, Q_i, \tag{3.11}$$

with $\alpha(P)$ denoting the area of the triangle or quadrangle P.

Let us turn to the investigation of Step 3. To allow the variation of the actual vertices in the direction of E_i, we define

$$\tilde{p}_i(s_i) = m(E_i) + \frac{s_i}{2} E_i, \quad s_i \in \mathbb{R}, \quad i = 0, \ldots, n \tag{3.12}$$

and

$$\tilde{e}_i(s_{i-1}, s_i) = [\tilde{p}_{i-1}(s_{i-1}), \tilde{p}_i(s_i)] \,, \quad i = 1, \ldots, n \,. \tag{3.13}$$

Note that $p_i = \tilde{p}_i(0)$ and $e_i = \tilde{e}_i(0,0)$ are recovered for $s_i = 0$. Obviously the points $\tilde{p}_i$, $i = 0, \ldots, n$, satisfy (3.8) if and only if

$$|s_i| \leq 1 \,, \quad i = 0, \ldots, n \tag{3.14}$$

holds. On the other hand, simple calculation shows that the condition (3.9) applied to $\tilde{e}_i$ reduces to the linear system

$$a_i s_i - b_{i-1} s_{i-1} = c_i, \quad i = 1, \ldots, n \tag{3.15}$$

for the unknowns s_i with coefficients a_i, b_{i-1} given by

$$a_i = \tfrac{1}{2}(E_i, G_i) \,, \; b_{i-1} = \tfrac{1}{2}(E_{i-1}, G_i) \,, \; c_i = -(e_i, G_i) \,, \; i = 1, \ldots, n. \tag{3.16}$$

Finally (3.10) can be written as

$$\sum_{i=0}^{n} s_i^2 = \min \,. \tag{3.17}$$

For the moment, let us neglect the restriction (3.14) and consider the unrestricted problem which consists only of the system (3.15) together with the additional condition (3.17).

Proposition 3.6 Assume that the condition

$$a_i \neq 0\ ,\quad i = 1, \ldots, n \tag{3.18}$$

holds. Then the unique solution of the unrestricted problem is given by

$$s_i^* = s_i^{(0)} + \lambda s_i^{(1)}\ ,\quad i = 0, \ldots, n \tag{3.19}$$

where

$$\lambda = \left(\sum_{i=0}^{n} s_i^{(0)} s_i^{(1)}\right) \Big/ \left(\sum_{i=0}^{n} (s_i^{(1)})^2\right) \tag{3.20}$$

and $s_i^{(0)}, s_i^{(1)},\ i = 0, \ldots, n$, given recursively by

$$s_0^{(0)} = 0,\ s_i^{(0)} = \frac{1}{a_i}(b_{i-1} s_{i-1}^{(0)} + c_i),\quad i = 1, \ldots, n \tag{3.21}$$

$$s_0^{(1)} = 1,\ s_i^{(1)} = \frac{1}{a_i} b_{i-1} s_{i-1}^{(1)},\quad i = 1, \ldots, n\ . \tag{3.22}$$

Proof: Obviously $s^{(0)} = (s_0^{(0)}, \ldots, s_n^{(0)})$ and $s^{(1)} = (s_0^{(1)}, \ldots, s_n^{(1)})$ defined by (3.21) and (3.22) are particular and homogeneous solutions of (3.15), respectively. As a consequence of (3.18) the solution space of (3.15) has dimension one, yielding the representation (3.19). Finally (3.20) follows easily from (3.17).

Remark 3.7 Let $a_j = (E_j, G_j) = 0$ for some fixed $j = 1, \ldots, n$ violating the condition (3.18). Then the edge E_j which has been constructed to intersect the numerical layer, turns out to be parallel to it. Hence the violation of (3.18) indicates that the discrete layer γ_k has not been constructed correctly.

As the solution $s_i^*, i = 0, \ldots, n$ of the unrestricted problem is uniquely determined, the restricted problem admits no solution, if $|s_i^*| > 1$ for some $i = 1, \ldots, n$, with $s^* = (s_0^*, \ldots, s_n^*)$ computed from (3.19).
Moreover, the case $s_i = 1$ has to be prohibited replacing (3.14) by the stronger condition

$$\max_{i=0,\ldots,n} |s_i| \leq \nu < 1\ , \tag{3.23}$$

with the parameter $\nu \geq 0$ limiting the degeneration of triangles with vertex $p_i, i = 0, \ldots, n$. On the other hand, too small corrections are not worth to be carried out and might even deteriorate the results, so that

$$\max_{i=0,\ldots,n} \|s_i E_i\| \geq \mu > 0 \tag{3.24}$$

is additionally required. In our numerical experiments we have used $\nu = 0.8$ and $\mu = 0.001$.

Let us summarize in detail how to perform Step 3 of Algorithm 3.3 .

The two grid orientation failed, if one of the conditions (3.18) or (3.23) is violated. Else the new vertices $p_i^*, i = 0, \ldots, n$ are given by

$$p_i^* = \begin{cases} \tilde{p}_i(s_i^*) & \text{if (3.24) holds} \\ p_i & \text{else} \end{cases} \qquad i = 0, \ldots, n,$$

where the displacements $s_i^*, i = 0, \ldots, n$, are computed according to (3.19).

The final Step 4 in Algorithm 3.4 is easily performed by onedimensional interpolation on the edges $E_i, i = 1, \ldots, n$.

Let us now come back to the crucial condition (3.23).

Obviously the violation of (3.23) indicates that the preceding triangulation $\mathcal{T}_{k-1}$ is not sufficiently adapted to the approximate solution U_k. Hence the orientation of $\mathcal{T}_k$ has to be be prepared by the orientation of $\mathcal{T}_{k-1}$. In this way we may descend to some coarsest level $r_k \geq 1$. Choosing $r_k = 1$ in the first orientation step the condition (3.23) is satisfied, if the initial triangulation $\mathcal{T}_0$ only roughly reflects the behaviour of U_k. In most practical applications the choice of $\mathcal{T}_0$ may be additionally supported by physical considerations.

This multilevel approach requires the restriction of the refinement direction $g(t), t \in \mathcal{T}_k$ to lower levels and an interpolation of the approximate solution U_k. Clearly the restriction of g to levels $j < k$ can be performed by successive application of Step 2 of Algorithm 3.3. Let us briefly discuss the interpolation of U_k to $\mathcal{T}_k^*$. Here the deformation of triangles $T \in \mathcal{T}_j$ resulting from the orientation on some level $j < k$ is inherited by the sons of these triangles which belong to the triangulations $\mathcal{T}_{j+1}, \ldots, \mathcal{T}_k$. Hence the orientation of $\mathcal{T}_j$ will produce a whole sequence of new triangulations $\mathcal{T}_j^*, \mathcal{T}_{j+1}', \ldots, \mathcal{T}_k'$. As the displaced nodes $\mathcal{P}_k' \backslash (\mathcal{P}_k' \cap \mathcal{P}_k)$ cannot be located as easily as in the two level case, the interpolation of U_k from $\mathcal{T}_k$ to $\mathcal{T}_k'$ and finally to $\mathcal{T}_k^*$ becomes more complicated.

The following multilevel grid orientation provides the oriented triangulation $\mathcal{T}_k^*$ together with the oriented discrete layer $\gamma_k^* = (e_i^*)_{i=0}^n$ on level k.

Algorithm 3.8: Multilevel grid orientation

Step 1: In the first multilevel orientation step choose $r_k = 1$. In future steps r_k is equal to the coarsest level on which the triangulation has been modified in the preceding multilevel orientation.

Step 2: Initialize $j := r_k, \tilde{U}_k := U_k$ and $\tilde{\mathcal{T}}_i := \mathcal{T}_i$ for $i = r_k, \ldots, k$.

Step 3: Compute the refinement directions $\tilde{g}(t) = \operatorname{grad} \tilde{U}_k|_t, t \in \tilde{\mathcal{T}}_k$, and restrict them to the coarse triangles $T \in \tilde{\mathcal{T}}_j$.

Step 4: Perform a two level orientation of $\tilde{\mathcal{T}}_j$ according to Algorithm 3.3 , changing the triangulations $\tilde{\mathcal{T}}_j, \tilde{\mathcal{T}}_{j+1}, \ldots, \tilde{\mathcal{T}}_k$ into $\tilde{\mathcal{T}}'_j, \tilde{\mathcal{T}}'_{j+1}, \ldots, \tilde{\mathcal{T}}'_k$.

Step 5: Compute $\tilde{U}'_k \in \mathcal{S}(\tilde{\mathcal{T}}'_k)$ by interpolation of $\tilde{U}_k \in \mathcal{S}(\tilde{\mathcal{T}}_k)$.

Step 6: If $j = k$ set $\mathcal{T}_k^* := \tilde{\mathcal{T}}'_k, U_k^* := \tilde{U}'_k$ and stop. Else set $j := j + 1, \tilde{\mathcal{T}}_i := \tilde{\mathcal{T}}'_i, i = j, \ldots, k$, $\tilde{U}_k := \tilde{U}'_k$ and continue with Step 3.

In case the two level orientation fails and $r_k > 1$, we may restart with Step 2 and $r_k := r_k - 1$. If Step 4 fails with $r_k = 1$, the algorithm should notify the user to choose $\mathcal{T}_0$ more appropriately and stop.

Remark 3.9 In most of all practical applications, the direction of the layer is recognized on comparatively low levels, while much finer triangulations are needed to obtain sufficient accuracy of the approximate solutions. Hence the number $k - r_k$ of levels touched by the multilevel orientation, decreases with increasing k. On higher levels no orientation is performed any more in view of (3.24) .

4. Anisotropic refinement

We are now ready to compile an anisotropic refinement algorithm which is especially designed for the efficient resolution of internal or boundary layers. Starting with some given initial triangulation $\mathcal{T}_0$ we perform $k_0 > 0$ uniform red refinements in advance of the first anisotropic refinement step. Note that $\mathcal{T}_0$ is not affected by the following orientation and hence should be chosen very coarse. In view of Definition 3.1, k_0 must be greater than zero. Assuming that a hierarchy of triangulations $\mathcal{T}_0, \ldots, \mathcal{T}_k, k \geq k_0$, has been constructed and that an approximate solution U_k has been computed on $\mathcal{T}_k$, anisotropic refinement is performed as follows.

Algorithm 4.1: Anisotropic refinement

Step 1: Compute an oriented triangulation $\mathcal{T}_k^*$ together with an oriented discrete layer $\gamma_k^* = (e_i^*)_{i=1}^n$, according to Algorithm 3.8.

Step 2: Skip all green refinements.

Step 3: Mark a subset of edges $\mathcal{E}_k^\# \subset \mathcal{E}_k$ for refinement.

Step 4: Search for pairs for possible blue refinement. Start with the triangles neighbouring e_i^*, $i = 1, \ldots n$, and continue in actual refinement direction.

Step 5: Mark the remaining triangles $t \in \mathcal{T}_k$ neighbouring an edge $e \in \mathcal{E}_k^\#$ for refinement.

Step 6: Apply the red refinement to all marked triangles.

Step 7: Continue the red refinement of all triangles $t \in \mathcal{T}_k$ which either have more than one refined neighbour, or which have exactly one neighbour but are not allowed for green refinement according to (G).

Step 8: Apply the green refinement to all triangles with one refined neighbour.

If no layer is present, Step 1 and Step 4 are skipped to reduce the algorithm to the isotropic refinement presented in the first chapter.

5. Implementation

The adaptive finite element code Kaskade described in [19] is used as a basis for an implementation of the anisotropic refinement strategy. For this purpose the underlying data structures developed by Leinen [16] were extended in an appropriate way.

Basic types. The objects of a triangulation, i.e. points, edges and triangles are stored as compact data (records, structures – the nomenclature depends on the programming language). These objects are identified as a whole through one identifier (pointer, address, index) which allows the reference to the elements of the objects.

The data types of the basic objects are collected in Tables 5.1, 5.2 and 5.3.

`x,y`	x, y–coordinate
`boundP`	boundary type descriptor
`next`	identifier of the next point
`last`	identifier of the previous edge
`level`	number of refinement level
`indexP`	number of point
`vector`	associated array of real numbers

Table 5.1 Data type for the point object

`p1,p2`	identifier of end points
`[pm]`	identifier of midpoints
`t1,t2`	identifier of neighbour triangles
`boundP`	boundary type descriptor
`next`	identifier of the next edge
`last`	identifier of the previous edge
`father`	identifier of the father edge
`firstSon`	identifier of the first son edge
`level`	refinement level
`type`	refinement type descriptor
`vector`	associated array of real numbers

Table 5.2 Data type for the edge object

e1,e2,e3	identifier of the edges
[p1,p2,p3]	identifier of the points
next	identifier of the next triangle
last	identifier of the previous triangle
father	identifier of the father triangle
son	identifier of the first son triangle
level	refinement level
type	refinement type descriptor
vector	associated array of real numbers

Table 5.3 Data type for the triangle object

The first group of elements is necessary to define the object, the second contains structural information (and should be hidden to the user), the third is the storage for the actual computation. Redundant elements are marked by brackets, they are included for efficiency reasons or programming convenience.

The elements of a data type are accessed by "functions" which might be (depending on the implementation) directly accessed as defined in some programming languages or indirectly by macros or functions.

Triangulations. The data types for points, edges and triangles are combined in the triangulation data structure which is used by the following groups of functions

- application of user functions on sets of points, edges or triangles (see Table 5.4);
- procedures to refine the current triangulation adaptively and
- procedures to create and delete complete triangulations.

The actual algorithms used to implement these functions are hidden to the user. In our implementation we use double linked lists and (for edges and triangles) trees as indicated by the structural information in Tables 5.1, 5.2 and 5.3.

Invariants. The actual refinement process is done in a systematic way. Therefore a set of additional rules for the data types or their relationship hold:

ApplyP	apply a function on all points selected	
	all	select all points
	initial	select all points of the initial triangulation
	dirichlet	select all points on the boundary with Dirichlet boundary condition
	list	select all points in list list
ApplyE	apply a function on all edges selected	
ApplyT	apply a function on all triangles selected	
ApplyLevel	apply a function on all points, edges or triangles of a given refinement level	

Table 5.4 Some functions to access the triangulation data structure

- The fields p1, p2, t1, t2 of an edge and p1, p2, p3, e1, e2, e3 of a triangle are never changed.
- Edges are refined in a natural way, the direction is maintained. This means that

```
ed->firstSon->p1  =  ed->p1
ed->firstSon->p2  =  ed->pm
```

  holds for a refined edge ed.
- The sequence of points p1, p2, p3 and edges e1, e2, e3 of a triangle are mathematical positive oriented. p1 lies opposite e1 and so on.
- The fourth son of a red triangle triangle is the middle triangle. The first son is near p1 and so on.
- Rules exist for blue refinement allowing the straight forward access to the inner or the opposite edge.

To clarify this approach we give some examples of the usage of the triangulation data structure.

Example 5.1 This example shows the red refinement of selected triangles occurring in Steps 2, 3 of Algorithm 1.1 and Steps 5, 6 of Algorithm 4.1, respectively. A triangle is refined (in the "red" fashion), if it carries a quantity (s) greater than a certain threshold (theta, a global variable) in one of its edges . These numbers may be computed by an error estimator or some related procedure.

```
global real theta;

integer function RefLocal(e)
   edge identifier e;
   if(RA(e,s)>theta)
      RefRed(e->t1);RefRed(e->t2);
   endif;
   return takeNext;
endfunction;

procedure RefAdaptively()
   OpenForRefinement();
   ApplyE(RefLocal,all);
   CloseRefinement();
endprocedure;
```

The procedure `RefRed` marks a triangle to be refined. The actual refinement is invoked by calling `CloseRefinement`. The macro `RA` is used to access the associated array. The constant `takeNext` signals `ApplyT` to take the next triangle of the triangulation.

Example 5.2 This example demonstrates the recursive approach to find the triangle which contains a given point p. It may be used in Step 5 of Algorithm 2.8 . First we search the triangle $T \in \mathcal{T}_0$ which is including p. Then the sons of T are tested recursively. Again we have to consider the whole quadrangle $t \cup t'$ in case of blue refinement of some triangle t. Note that `ApplyT` stops the process if `TestTriangle` returns `stopApply`.

```
global point identifier lookForP;
global triangle identifier includesP;

integer function TestTriangle(t)
   triangle identifier t;
   if (Includes(lookForP, t))
      if (firstSon(t)=nil)
         includesP=t;
         return stopApply;
      endif;
      if (TestTriangle(SON1(t))=stopApply)
        return stopApply; endif;
      if (TestTriangle(SON2(t))=stopApply)
        return stopApply; endif;
```

```
      if (type(t)=blue)
         if (TestTriangle(BLUENEIGHBOR(t))=stopApply)
           return stopApply; endif;
       endif;

      if (type(t)=green)
        return takeNext; endif;

      if (TestTriangle(SON3(t))=stopApply)
        return stopApply; endif;
      if (TestTriangle(SON4(t))=stopApply)
        return stopApply; endif;
   endif;
   return takeNext;
endfunction;

triangle identifier function FindTriangle(p)
   point identifier p;
   lookForP := p;
   includesP := nil;
   ApplyT (TestTriangle, initial);
   return includesP;
endfunction;
```

The function `Includes` checks, if `p` lays within the boundary of the triangle `t`. The macros `firstSon` and `type` denote access to the corresponding fields of the data type for triangles, `blue` and `green` are constants for `type(t)` which is set if `t` was refined in the blue or green fashion. `SON1` to `SON4` are macros to access the identifiers of the sons of a triangle. `BLUENEIGHBOR(t)` gives the triangle t' to cover the complete quadrangle.

6. Numerical Results

In this chapter the anisotropic refinement strategy developed above is applied to the resolution of internal and boundary layers generated by the simple convection diffusion equation

$$\begin{aligned} -\varepsilon\Delta u + \beta \cdot \operatorname{grad} u &= f \quad \text{on } \Omega = (0,1)\times(0,1)\,, \\ u|_{\Gamma_0} = u_0 \;,\; \frac{\partial u}{\partial n}|_{\Gamma_1} &= 0 \quad \text{on } \partial\Omega = \Gamma_0 \cup \Gamma_1,\ \Gamma_0 \cap \Gamma_1 = \emptyset \end{aligned} \tag{6.1}$$

with $\beta \cdot \operatorname{grad} u = \beta_x \dfrac{\partial}{\partial x} u + \beta_y \dfrac{\partial}{\partial y} u$.

We will shortly sketch an appropriate finite element discretization of (6.1).

Let $\mathcal{T}$ be a triangulation of Ω. Then the standard Galerkin method for the approximate solution of (6.1) reads as follows.

Find $U \in S(\mathcal{T})$ with the property $U|_{\Gamma_0} = u_0$ so that

$$\varepsilon(\operatorname{grad} U, \operatorname{grad} v) + (\beta \cdot \operatorname{grad} U, v) = (f, v) \tag{6.2}$$

holds for all $v \in S(\mathcal{T})$ satisfying $v|_{\Gamma_0} = 0$.

For details we refer to Ciarlet [7], Johnson [14] or any other book on finite elements.

If $0 < \varepsilon \ll 1$ and $\|\beta\| = (\beta_x^2 + \beta_y^2)^{1/2} \approx 1$, the convection term $\beta \cdot \operatorname{grad} u$ dominates the Laplacian so that internal or boundary layers may occur in the solution u of (6.1). Now it is well–known that the standard Galerkin method (6.2) produces non–physical oscillations where the solution u is non–smooth as long as $\mathcal{T}$ is not excessively fine. In view of the physical domain of dependence of the nodal points, stabilizing artificial diffusion should act only in the flux direction β. This reasoning leads to the following streamline diffusion method introduced by Hughes et al. in [12].

Find $U \in S(\mathcal{T})$ with the property $U|_{\Gamma_0} = u_0$ so that

$$\varepsilon(\operatorname{grad} U, \operatorname{grad} v) + (\beta \cdot \operatorname{grad} U, v + \delta\beta \cdot \operatorname{grad} v) = (f, v + \delta\beta \cdot \operatorname{grad} v) \tag{6.3}$$

holds for all $v \in S(\mathcal{T})$ satisfying $v|_{\Gamma_0} = 0$.

Here $\delta : \Omega \to \mathbb{R}$ is a positive, piecewise constant function, modeling the local step–size in flux direction. The definition

$$\delta(t) = \text{diameter of } t \text{ in direction } \beta\,, \quad t \in \mathcal{T} \tag{6.4}$$

will be used in the sequel.

It has been shown by Johnson et al. that the streamline diffusion method combines good stability properties with almost optimal accuracy. Nevertheless the resulting scheme is not monotone so that oscillations may still occur.

This may be remedied by an additional shock capturing term described in [18] which on the other hand tends to smear out sharp fronts or jumps. For details we refer to the monograph of Johnson [14] and the bibliography therein. As it turns out that stability may be also achieved by adaptively oriented grids, the streamline diffusion method (6.2) without shock capturing will be used in the following numerical examples

The application of a finite element method to the problem (6.1) requires the solution of a nonsymmetric linear system. For this purpose, we use the symmetric Gauß–Seidel iteration with relaxation parameter $\omega = 0.6$. The iteration is stopped, if the residual is less than 10^{-5}.

Finally the anisotropic refinement of some triangulation $\mathcal{T}_k$ requires the selection of a subset $\mathcal{E}_k^{\#} \subset \mathcal{E}_k$ of edges to be subdivided. In order to fix a simple test environment, these edges are determined simply by the gradient of the corresponding approximate solution U_k. To be precise, let

$$s(e) = \frac{| U_k(p_1) - U_k(p_2) |}{\|p_1 - p_2\|}, \quad e = (p_1, p_2) \in \mathcal{E}_k \tag{6.5}$$

and

$$S = \sum_{e \in \mathcal{E}_k} s(e) |\mathcal{E}_k|^{-1} \tag{6.6}$$

with $|\mathcal{E}_k|$ denoting the number of elements of $\mathcal{E}_k$. Then an edge $e \in \mathcal{E}_k$ is marked for refinement, if

$$s(e) \geq \theta := \zeta S \tag{6.7}$$

with some constant $\zeta > 0$. We will use $\zeta = 1.5$ in the sequel.

Example 6.1: Boundary layers. In many problems of practical interest the appearance of boundary layers is known in advance. In this case the direction of a layer γ is given a priori by the geometry of Ω and we may choose an initial triangulation $\mathcal{T}_0$ so that blue refinement can be applied directly without a preceding orientation of the actual triangulation.

As a simple example we chose problem (6.1) with $\varepsilon = 10^{-5}$, $\beta = (0, 1)$, $f \equiv 1$ and homogeneous Dirichlet boundary conditions. It is easily seen that the exact solution exhibits an ordinary boundary layer at the outflow boundary $\Gamma_{\text{out}} = \{(x, y) \in \partial\Omega | y = 1\}$ and the birth of a parabolic layer at the characteristic boundary $\Gamma_{\text{char}} = \{(x, y) \in \partial\Omega | 0 < y < 1\}$.

We start with the initial triangulation $\mathcal{T}_0$ illustrated in Figure 6.1 .

The first approximation U_2 is computed on the triangulation $\mathcal{T}_2$ resulting from $k_0 = 2$ uniform red refinements of $\mathcal{T}_0$. Figure 6.2 shows $\mathcal{T}_2$ together with the level curves of U_2.

To prepare blue refinement a discrete layer γ_2 is constructed according to Algorithm 2.3. The resulting polygonal is marked by a dotted line in Fig-

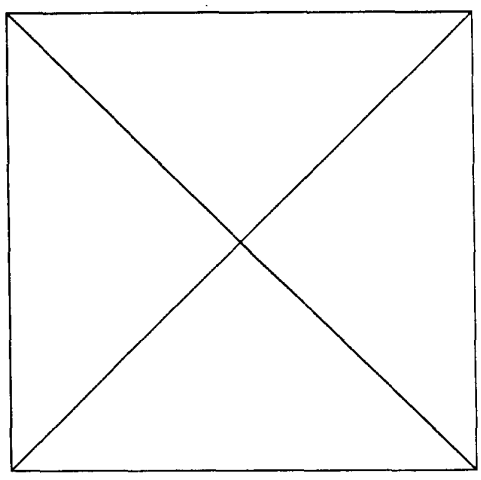


Figure 6.1 Initial Triangulation $\mathcal{T}_0$

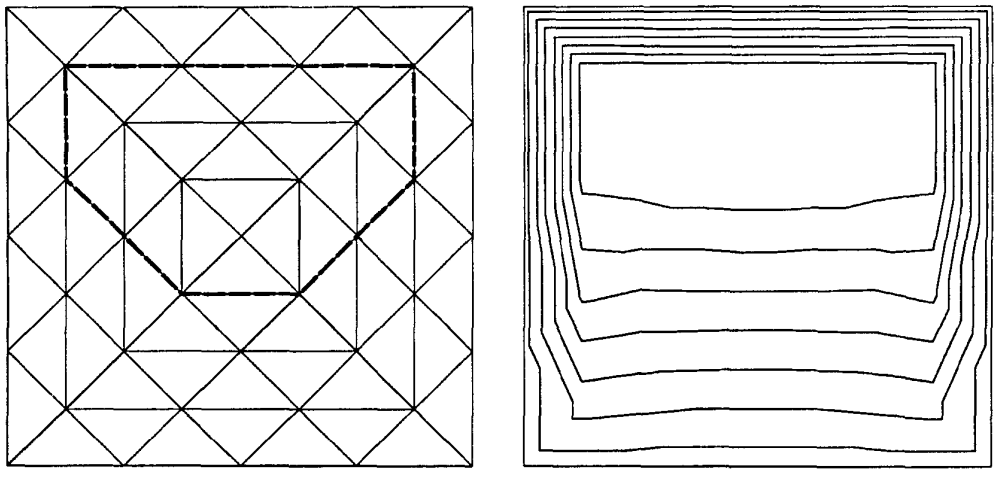


Figure 6.2 $\mathcal{T}_2$ with level curves of U_2

ure 6.2 and only partly coincides with the expected result. Though better performance is obtained on higher levels, this example confirms that a more reliable way of detecting discrete layers should be developed.

Now blue refinement takes place in the neighbourhood of γ_2 based on the conditions (B1) - (B4) stated in the second chapter. The resulting triangulation $\mathcal{T}_3$ together with the level curves of U_3 is shown in Figure 6.3.

Watch the forgotten triangle at the outflow boundary which is lacking a

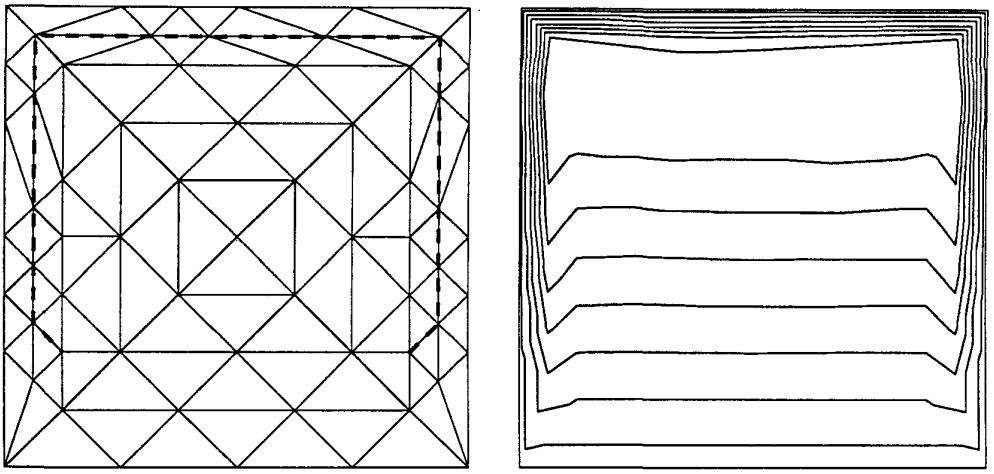


Figure 6.3 $\mathcal{T}_3$ with level curves of U_3

blue partner and hence remains for red refinement. Again the dotted line represents the discrete layer γ_3 which is now in better accordance with our expectations.

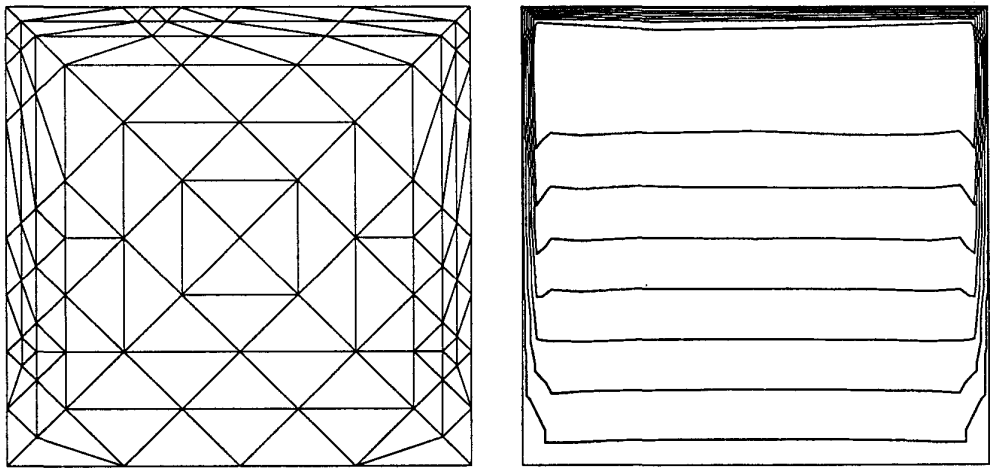


Figure 6.4 $\mathcal{T}_4$ with level curves of U_4

The situation after one further anisotropic refinement step is illustrated in Figure 6.4. Due to the angle condition (B4) successive blue refinement is performed in a natural way. Note that the red refined triangle indicated above will reproduce similar situations on higher levels. This effect, which

may be weakened by a more sophisticated initial triangulation, somewhat disturbs the beauty of the resulting triangulations but is of minor importance from the computational point of view.

We continued our calculations up to level 7. Figure 6.5 shows the final triangulation $\mathcal{T}_7$ together with a zoom of the situation close to the left upper edge including the level curves of U_7.

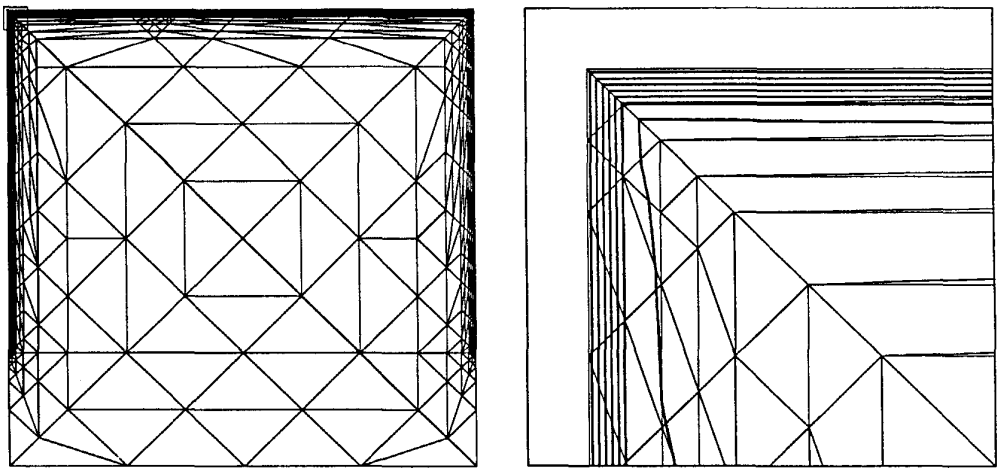


Figure 6.5 Triangulation $\mathcal{T}_7$ (347 nodes) and zoom.

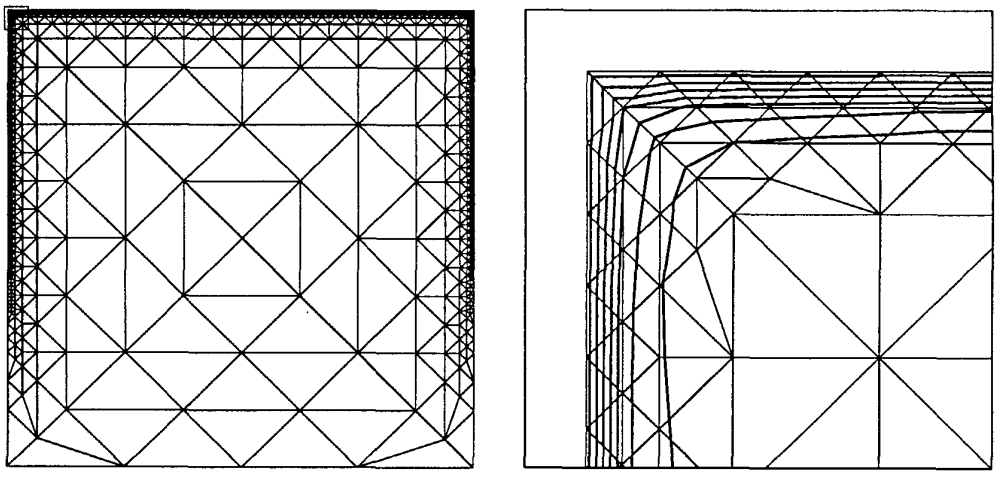


Figure 6.6 Triangulation $\bar{\mathcal{T}}_7$ (1178 nodes) and zoom.

For comparison we repeated the calculations, this time using simple isotropic

refinement according to Algorithm 1.1. The corresponding results are shown in Figure 6.6.

The resulting triangulation $\bar{\mathcal{T}}_7$ involves more than 3 times the number of nodes of $\mathcal{T}_7$ which allows even for a sharper resolution of the layer.

Example 6.2: Linear interior layer. Using our model problem (6.1) we produce a parabolic internal layer γ by transporting a discontinuity in the inflow condition across the computational domain Ω. In this example we keep the flux direction $\beta \equiv$ const to obtain a linear layer.

For actual computation we choose $\varepsilon = 10^{-5}$, $\beta = (1.0,\ 0.5)$, $f \equiv 0$, and

$$u_0(x,y) = \begin{cases} 0 & y > 0.3 \\ 1 & y \leq 0.3 \end{cases}, \quad (x,y) \in \Gamma_0$$

with $\Gamma_0 = \Gamma_{\text{in}} = \{(x,y) \in \partial\Omega | \max(x,y) < 1\}$ and $\Gamma_1 = \Gamma_{\text{out}} = \partial\Omega \backslash \Gamma_{\text{in}}$. It is easily seen that then the exact solution shows a linear internal layer proceeding from the discontinuity $(0.0, 0.3) \in \Gamma_{\text{in}}$ in the flux direction β to $(1.0, 0.8) \in \Gamma_{\text{out}}$.

The initial triangulation $\mathcal{T}_0$ is chosen as coarse as possible and displayed in Figure 6.7.

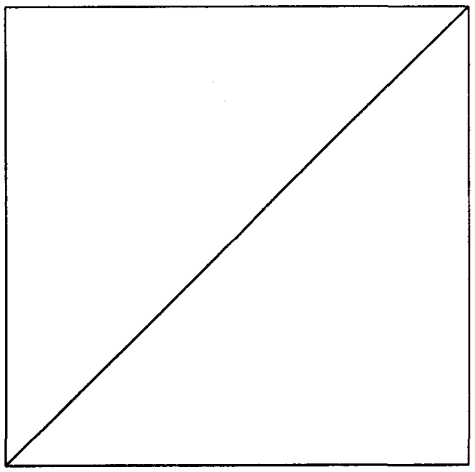


Figure 6.7 Initial triangulation $\mathcal{T}_0$.

To illustrate the orientation process we start our calculation on the triangulation $\mathcal{T}_1$, resulting from $k_0 = 1$ uniform refinement of $\mathcal{T}_0$. Figure 6.8 shows on the left the triangulation $\mathcal{T}_1$ together with the level curves of the approximation U_1. The discrete layer γ_1 detected by Algorithm 3.3 is marked by

a dotted line. On the right hand side we display the oriented triangulation $\mathcal{T}_1^*$ with the discrete layer γ_1^* adjusted perpendicular to the local refinement direction.

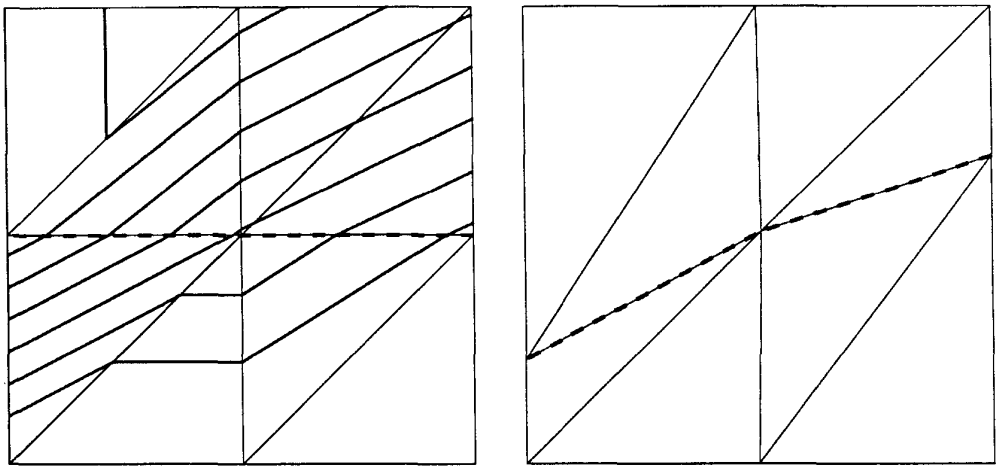


Figure 6.8 $\mathcal{T}_1$ with level curves of U_1 and oriented triangulation $\mathcal{T}_1^*$.

The corresponding situation after another uniform refinement step is shown in Figure 6.9. Of course we also might have started with two uniform refinements of $\mathcal{T}_0$ with very similar results.

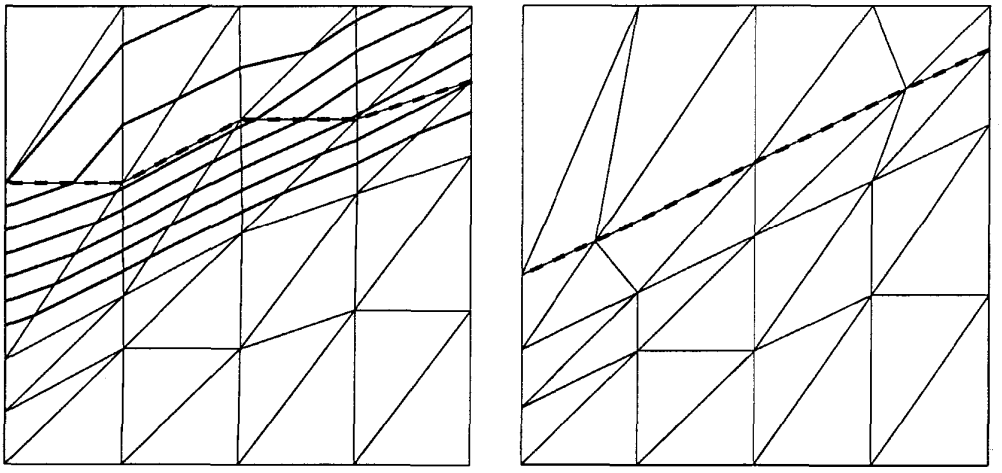


Figure 6.9 $\mathcal{T}_2$ with level curves of U_2 and oriented triangulation $\mathcal{T}_2^*$.

Now Figures 6.10 and 6.11 show the successive blue refinement close to the internal layer on the following levels 3 and 4. As a consequence of (3.24) no further orientation is performed on level 4.

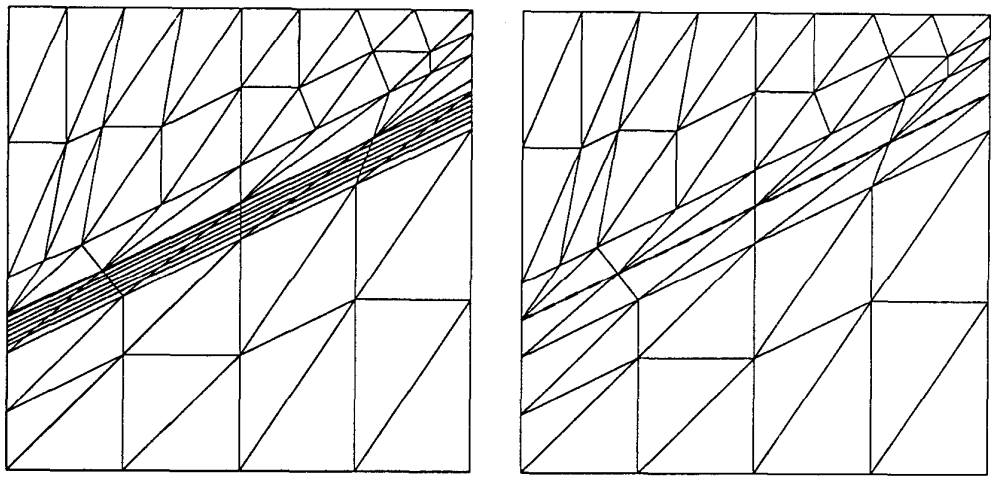


Figure 6.10 $\mathcal{T}_3$ with level curves of U_3 and oriented triangulation $\mathcal{T}_3^*$.

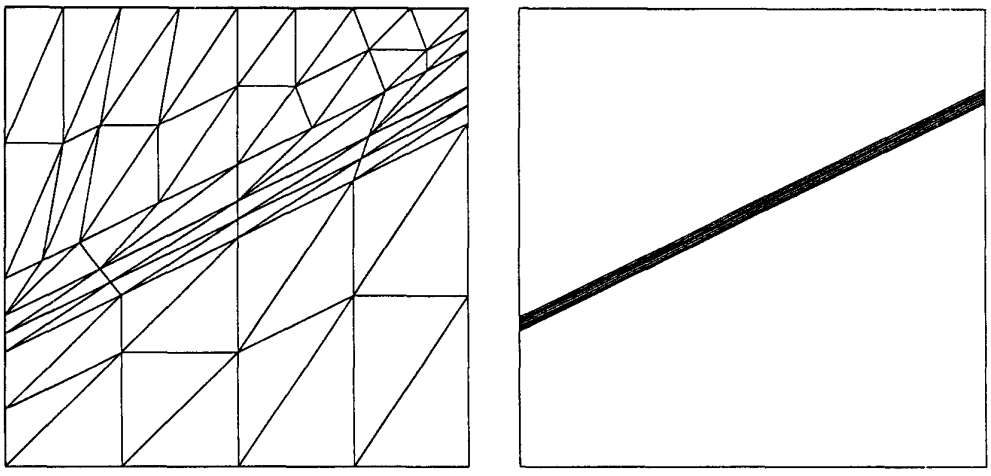


Figure 6.11 $\mathcal{T}_4$ with level curves of U_4.

Again anisotropic refinement is continued up to level 7. The final triangulation $\mathcal{T}_7$ together with the level curves of U_7 is shown in Figure 6.12.

The corresponding results based on simple isotropic refinement are illustrated in Figure 6.13.

Obviously the resolution of the layer is much worse though almost ten times more grid points are involved in the calculation. Furthermore a considerable stabilizing effect of the oriented triangulation is observed. Indeed by

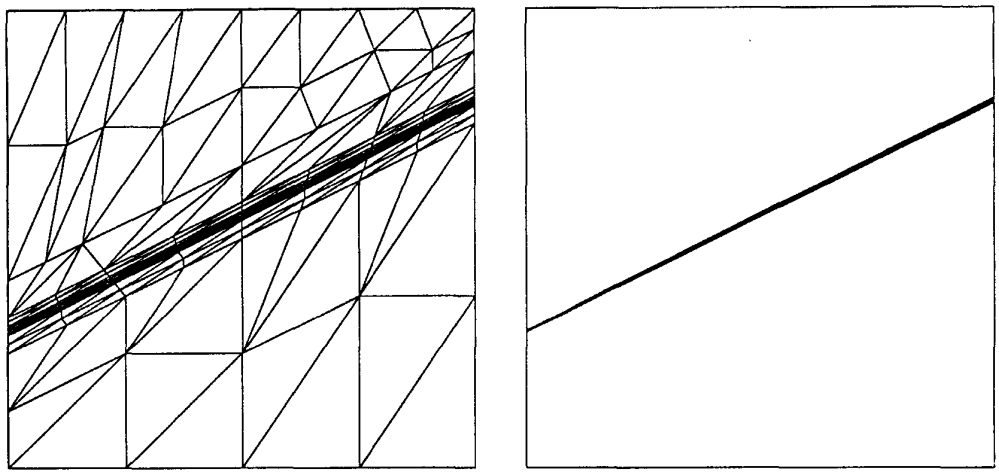


Figure 6.12 $\mathcal{T}_8$ (118 nodes) and level curves of U_7.

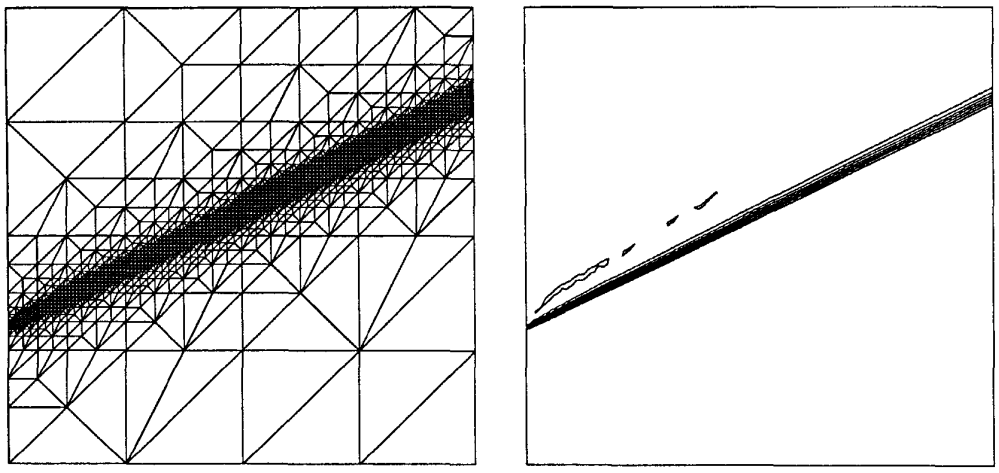


Figure 6.13 $\bar{\mathcal{T}}_8$ (1462 nodes) and level curves of $\bar{U}_8$.

the use of oriented grids the different phases of the solution are separated properly so that the physical domain of dependence is modeled correctly by the discretization. Following Courant, Friedrichs, and Lewy [8] this yields good stability of the method. In this simple case the resulting scheme may also be viewed as a local characteristic method which is well suited for the approximation of the dominant convection term.

Example 6.3: Curved interior layer. Let us change the flux direction to $\beta = (y, -x)$ and the boundary conditions to

$$u_0(x,y) = \begin{cases} 0 & y > 0.7 \\ 1 & y \leq 0.7 \end{cases}, \quad (x,y) \in \Gamma_0 ,$$

with $\Gamma_0 = \Gamma_{\text{in}} = \{(x,y) \in \partial\Omega | x = 0 \text{ or } y = 1\}$ and $\Gamma_1 = \Gamma_{\text{out}} = \partial\Omega \backslash \Gamma_{\text{in}}$. The other parameters are kept from the previous Example 6.2.

Again we use the initial triangulation $\mathcal{T}_0$ displayed in Figure 6.7

The first approximation U_3 is computed on the triangulation $\mathcal{T}_3$ resulting from $k_0 = 3$ uniform refinements of $\mathcal{T}_0$. Now Figure 6.14 shows the orientation of $\mathcal{T}_3$ emphasizing the multilevel structure of the algorithm. The dotted line in the right picture is showing the oriented discrete layer γ_3^* on the actual level.

The further orientation on level 4 is illustrated in Figure 6.15 .

The final triangulation $\mathcal{T}_7$ together with U_7 is shown in Figure 6.16 . As the local flux direction is not modeled exactly by the linear edges, the results are less optimal than in the preceding example. Still the usual isotropic refinement yields worse results at much more computational cost as follows from Figure 6.17.

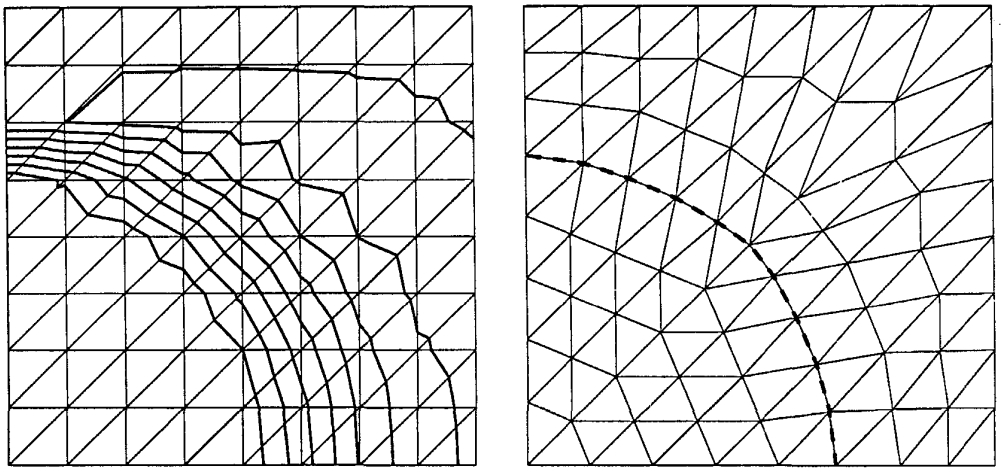


Figure 6.14 $\mathcal{T}_3$ with level curves of U_3 and the oriented triangulation $\mathcal{T}_3^*$.

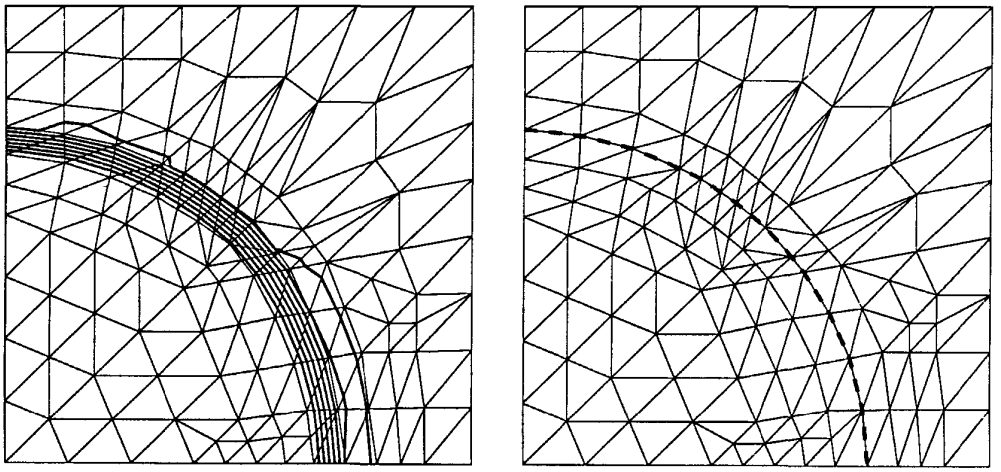


Figure 6.15 $\mathcal{T}_4$ with level curves of U_4 and the oriented triangulation $\mathcal{T}_4^*$.

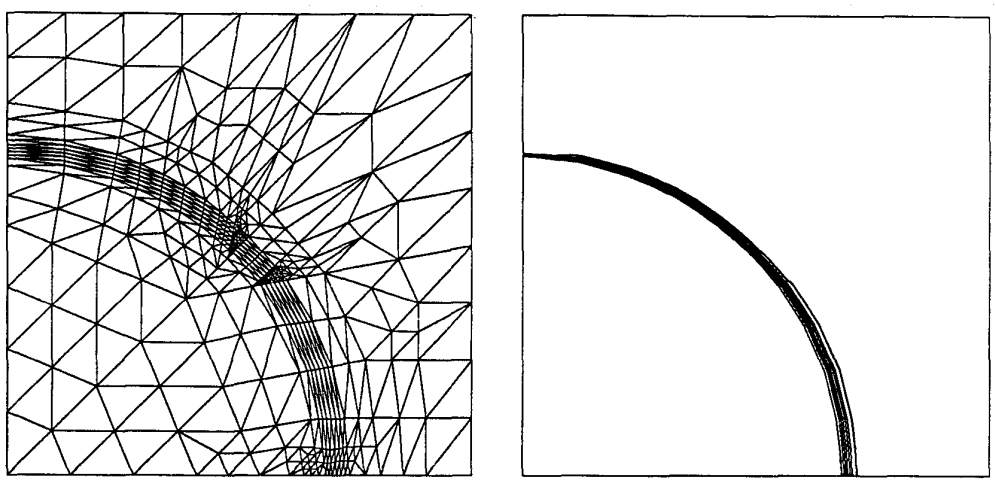


Figure 6.16 $\mathcal{T}_7$ (356 nodes) and level curves of U_7.

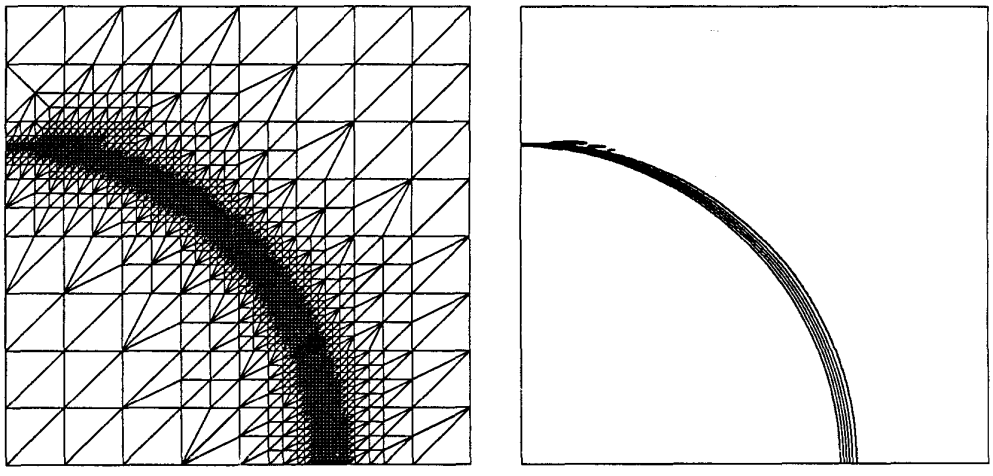


Figure 6.17 $\bar{\mathcal{T}}_7$ (1887 nodes) and level curves of $\bar{U}_7$.

References

[1] I. Babuška, A.K. Aziz: *On the Angle Condition in the Finite Element Method.* SIAM J. Num. Anal. Vol 13, No.2 p. 214–226 (1976).

[2] R.E. Bank: *PLTMG Users Guide, Edition 5.0.* Technical Report, Department of Mathematics, University of California at San Diego (1988).

[3] R.E. Bank, T.F. Dupont, H. Yserentant: *The Hierarchical Basis Multigrid Method.* Num. Math. 52, p. 427–458 (1988).

[4] R.E. Bank, A.H. Sherman, A. Weiser: *Refinement Algorithms and Data Structures for Regular Local Mesh Refinement.* Scientific Computing, R. Stepleman et al. (eds.), Amsterdam: IMACS North-Holland, p. 3–17 (1983).

[5] R.E. Bank, A. Weiser: *Some a-posteriori Error Estimators for Elliptic Partial Differential Equations.* Math. Comp. 44, p. 283–301 (1985).

[6] R.E. Bank, H. Yserentant: *Some Remarks on the Hierarchical Basis Multigrid Method.* To appear.

[7] P.G. Ciarlet: *The Finite Element Method for Elliptic Problems.* North-Holland, Amsterdam (1978).

[8] R. Courant, K.O. Friedrichs, H. Lewy: *Über die partiellen Differentialgleichungen der Physik.* Math. Ann. 100, p. 32–74 (1928).

[9] P. Deuflhard, P. Leinen, H. Yserentant: *Concepts of an Adaptive Hierarchical Finite Element Code.* IMPACT 1, p. 3–35 (1989).

[10] W. Eckhaus: *Asymptotic Analysis of Singular Perturbation Problems.* North-Holland Amsterdam, New York, Oxford (1979).

[11] W. Hackbusch: *Multi-Grid Methods and Applications.* Springer Verlag, Berlin, Heidelberg (1985).

[12] T.J.R. Hughes, A. Brooks: *A Multidimensional Upwind Scheme with no Crosswind Diffusion.* In: Finite Element Methods for Convection Dominated Flows. T.J.R. Hughes (ed.), AMD, Vol. 34 (ASME, New York) p. 19–35 (1979).

[13] P. Jamet: *Estimation d' erreur pour des éléments finis droits presque dégénérés.* R.A.I.R.O., Série Analyse Numérique, 10, No. 3, p. 43–61 (1976).

[14] C. Johnson: *Numerical Solutions of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, Cambridge (1987).

[15] R. Kornhuber, R. Roitzsch: *Adaptive Finite–Element–Methoden für Konvektionsdominierte Randwertprobleme.* Proceedings of the 4. TECFLAM–Seminar, Stuttgart p. 103–116 (1988).

[16] P. Leinen: Work done in preparation of a dissertation (1989).

[17] P.A. Markowich: *The Stationary Semiconductor Device Equations.* Springer Verlag Wien, New York (1986).

[18] A.Mizukami, T.J.R. Hughes: *A Petrov–Galerkin Finite Element Method for Convection–Dominated Flows: An Accurate Upwinding Technique for Satisfying the Maximum Principle.* Comput. Meths. Appl. Mech. Engrg. 50, p.181–193 (1985)

[19] R. Roitzsch: *KASKADE Programmer's Manual.* Technical Report TR89–5, Konrad–Zuse–Zentrum Berlin (1989).

[20] J.L. Synge: *The Hypercicle in Mathematical Physics.* Cambridge University Press, New York (1957).

[21] F. Thomasset: *Implementation of Finite Element Methods for Navier–Stokes Equations.* Springer Verlag New York, Heidelberg, Berlin (1981).

[22] M.I. Višik, L.A. Lyusternik: *Regular Degeneration and Boundary Layer for Linear Differential Equations with Small Parameter.* American Mathematical Society Translations, Series 2, Vol. 20 p. 239–364 (1962). Springer Verlag New York, Heidelberg, Berlin (1981).

[23] P. Wesseling: *Cell–Centered Multigrid for Interface Problems.* In: Multigrid Methods: Theory, Applications and Supercomputing. (Ed. S. McCormick), p. 631–641, Marcel Dekker, New York (1988).