Adaptive Anisotropic Meshing For Steady Convection-Dominated Problems

Hoa Nguyen [a] Max Gunzburger [b,1] Lili Ju [c,2] John Burkardt [d]

[a] *Mathematics Department, Tulane University, New Orleans, LA 70118, USA*

[b] *Department of Scientific Computing, Florida State University, Tallahassee, FL 32306-4120, USA*

[c] *Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA*

[d] *Interdisciplinary Center for Applied Mathematics, Virginia Tech, Blacksburg, VA 24061, USA*

Abstract

Obtaining accurate solutions for convection-diffusion equations is challenging due to the presence of layers when convection dominates the diffusion. To solve this problem, we design an adaptive meshing algorithm which optimizes the alignment of anisotropic meshes with the numerical solution. Three main ingredients are used. First, the streamline upwind Petrov-Galerkin method is used to produce a stabilized solution. Second, an adapted metric tensor is computed from the approximate solution. Third, optimized anisotropic meshes are generated from the computed metric tensor by an anisotropic centroidal Voronoi tessellation algorithm. Our algorithm is tested on a variety of two-dimensional examples and the results shows that the algorithm is robust in detecting layers and efficient in avoiding non-physical oscillations in the numerical approximation.

Key words: anisotropic mesh generation, metric tensor, convection-dominated problem, stabilized method

Email addresses: `hnguye4@tulane.ed` (Hoa Nguyen), `gunzburg@fsu.edu` (Max Gunzburger), `ju@math.sc.edu` (Lili Ju), `burkardt@vt.edu` (John Burkardt).

[1] Supported in part by DOE Office of Science Climate Change Prediction Program through grant number DE-FG02-07ER64432 and in part by National Science Foundation under grant number DMS-0308845.

[2] Supported in part by DOE Office of Science Climate Change Prediction Program through grant number DE-FG02-07ER64431 and in part by the National Science Foundation under grant number DMS-0609575.

1 Introduction

We are interested in constructing numerical methods for solving partial differential equations whose solutions contain steep boundary and interior layers. In these layers, the solution varies much more quickly in some directions than it does in others. However, it is not always known a priori where these layers are located or in which directions the solution varies most quickly. A numerical method should detect the layers automatically and be robust and efficient in resolving the layers without causing non-physical oscillations in the numerical approximations. In this paper, we consider, as a model problem, the singularly perturbed convection-diffusion equation in two dimensions:

$$\begin{cases} -a\triangle u + \mathbf{v}\cdot\nabla u = f & \text{in } \Omega \\ u = g & \text{on } \Gamma, \end{cases} \tag{1}$$

where $\Omega \subset \Re^2$ is a bounded polygonal domain with boundary Γ, $0 < a \leq 1$ is a given constant diffusion coefficient, $\mathbf{v}(\mathbf{x}) \in [W^{1,\infty}(\Omega)]^2$ is a given convective/velocity field, $f(\mathbf{x}) \in L^2(\Omega)$ is a given source function, and $g \in H^{\frac{1}{2}}(\Gamma)$ is a given boundary function. We use standard notations for Sobolev spaces; see, e.g., [1]. The boundary Γ can be divided into three parts:

inflow boundary	$\Gamma^- = \{\mathbf{x}\in\Gamma : \mathbf{v}\cdot\mathbf{n} < 0\}$
characteristic boundary	$\Gamma^0 = \{\mathbf{x}\in\Gamma : \mathbf{v}\cdot\mathbf{n} = 0\}$
outflow boundary	$\Gamma^+ = \{\mathbf{x}\in\Gamma : \mathbf{v}\cdot\mathbf{n} > 0\}$,

where $\mathbf{n}$ denotes the unit outward normal vector to Γ.

Problem (1) models the transport of a quantity u through diffusion and convection processes. It arises in many science and engineering problems including pollutant or heat transport in a flowing fluid, in which case u denotes the concentration of the pollutant or the temperature, respectively. Several other examples of the application of (1) are given in [29]. In addition, (1) also serves as a prototype for a wider class of problems in which diffusion and convection processes play a central role. In fact, in [29], the problem (1) is identified as the most widespread fundamental sub-problem in science and engineering.

Solving (1) becomes challenging when convection dominates diffusion, i.e., when $a \ll \|\mathbf{v}\|$. In such cases, the solution usually exhibits very thin layers across which the derivatives of the solution are large. The layers can be classified into four main types.

- *Regular boundary layers* – appear at the outflow boundary Γ^+ where the velocity field $\mathbf{v}$ is not parallel to Γ.

- *Parabolic boundary layers* – appear at the characteristic boundary Γ^0 where the velocity field $\mathbf{v}$ is parallel to Γ.
- *Corner boundary layers* – appear in the neighborhood of a corner of the domain Ω where two boundary layers intersect.
- *Interior layers* – appear due to discontinuities in the boundary data at the inflow boundary Γ^-; the discontinuities are propagated across the domain along the vector lines of the velocity field $\mathbf{v}$.

When convection dominates diffusion, the solution of (1) has two parts, a smooth one and a rapidly varying one; the latter part describes the type of the boundary layer [15]. Regular boundary layers have width of $O(a)$; this is due to the significant difference between solutions of (1) and of the reduced equation ((1) with $a = 0$) at points near the outflow boundary. On the other hand, parabolic boundary layers are thicker in that they have width of $O(\sqrt{a})$. Interior layers have properties similar to the parabolic boundary ones in that they also have width of $O(\sqrt{a})$.

In cases where the solution contains steep layers, standard Galerkin finite element methods or central finite volume methods yield inaccurate oscillatory results unless the mesh is fine enough (relative to the ratio $\frac{a}{\|\mathbf{v}\|}$). In one dimension, it has been proved [29] that for linear finite elements on a uniform grid, non-physical oscillations occur when the mesh Peclet number $Pe = \frac{\|\mathbf{v}\|h}{2a}$ is greater than unity. This situation, however, is still not fully understood in two or more dimensions [13,32]. The difficulties are partly due to the many partial differential equation issues related to the smoothness of the solution, the compatibility of the data, and the geometry of the domain.

One common approach to solving equation (1) is to use a stabilizing scheme to enhance the stability of a standard methods. Popular methods include upwinding [18], Galerkin least-squares [22], and streamline upwind Petrov-Galerkin (SUPG) [3] methods. If a simulation is effected using uniform meshes, a lot of computational work is required but, in some cases, oscillations still appear near the layers [24]. As a result, nonuniform meshes such as those referred to as Bakhvalov and Shishkin type meshes [15] have been developed to adapt to the layers and improve the numerical approximation. However, the construction of the B- and S-meshes requires a priori knowledge of the locations and structures of the layers. In practice, it is impossible to know about all the non-smooth behavior of the solution. Hence, methods that adaptively generate meshes to capture the layers are very substantial interest.

In adaptive mesh generation, a posteriori error estimators are used to detect regions of large error for subsequent mesh refinement [2,23,36]. The refined meshes can be isotropic (regularly-shaped elements) or anisotropic (stretched elements). However, despite the advantage of reducing the computational cost, there are problems with using an a posteriori error estimator for equations of the type (1). First, it is difficult to achieve both the upper and lower bounds predicted by the error estimator; moreover, the error estimator gives no indication of the quality of the mesh. Consequently, non-physical oscillations still exist and convergence rates are low.

Most recent efforts directed at constructing adaptive anisotropic meshes are based on a stabilized scheme and some mesh modification strategy [30]. Such efforts are still in the very early stages of development. According to the survey article [34], there has been no adaptive anisotropic mesh generation method for the two-dimensional convection-diffusion equations that comes with a guaranteed bound for the error in the computed solution.

The method introduced in this paper attempts, in the context of convection-dominated problems, to robustly, efficiently, and automatically detect layers and produce accurate approximate solutions that do not exhibit non-physical oscillations. The construction of the method is based on three observations.

- A good choice of stabilized method can lead to small discretization errors and rapid convergence.
- An optimized anisotropic meshing scheme can reduce the computational cost and improve the accuracy of the solution by not only placing more nodes in layer regions, but by also distributing them in such a way that mesh elements are stretched along the layers.
- The stabilized method and the optimized meshing scheme can be tied together by a metric tensor that can be determined from the approximate solution of the stabilized method and then used to generate the mesh.

This motivates us to efficiently construct an adaptive algorithm that involves three main ingredients: a good stabilized method, a reliable metric tensor, and an optimized meshing scheme. The resulting algorithm effectively and robustly solves (1) on a general domain and automatically deals with different types of layers present in a solution. Here, we briefly discuss the choices we make for the three ingredients.

Stabilized scheme – Standard Galerkin methods are unstable for convection-diffusion equation when convection dominates diffusion. We choose to use the streamline upwind Petrov-Galerkin (SUPG) method [3]. Even though the SUPG is not monotone (i.e., it does not satify a discrete maximum principle), it is globally stable and has good higher-order accuracy in regions where the solution is smooth. Furthermore, the SUPG method is easy to implement and does not require higher-order or complicated weighting functions. SUPG adds extra diffusion to a standard discretization, but only along the streamlines, i.e., along directions parallel to $\mathbf{v}$. However, finding the optimal stabilization terms to completely diminish non-physical oscillations is still an open problem. Therefore, in practice, it might be impossible to achieve optimal convergence rates (i.e., second-order convergence for the L^2-error and first-order convergence for the H^1 error when using linear elements) if the solution possesses steep layers. A recent trend is to improve the convergence rate by adapting the mesh using approximate solutions of the stabilized scheme [8].

Metric tensor – Adaptive anisotropic mesh refinement has the advantage of improving the accuracy of the solution (particularly in layer regions) while reducing computation cost relative to methods using isotropic meshes. A suitable anisotropic mesh has to satisfy two principles: alignment and equi-distribution. Alignment guarantees that mesh elements are aligned with the geometry of the solution and can have large aspect ratio. Equidistribution forces the estimated error to be distributed uniformly over the mesh elements. These two principles are important in mesh adaptation to control the size, shape, and orientation of mesh elements. In our adaptive algorithm, we use a metric tensor to serve as a guide to align the mesh with the anisotropy of the computed solution. Metric tensors that are useful for anisotropic g rid generation are discussed in, e.g., [4–6,11,20,**?**]. We use a metric tensor developed in [20], mainly because it is derived by minimizing the upper bound of the interpolation error on a mesh satisfying the equi-distribution and alignment conditions. The metric tensor is determined from the approximate solution (more to the point, from the Hessian matrix of the computed solution) obtained on a mesh and is then used to define a metric from which a new mesh can be constructed.

Anisotropic mesh generator – Several anisotropic mesh generators in two dimensions are available for research purposes. Two examples are the BL2D [28] and the BAMG [17] packages that create two-dimensional isotropic or anisotropic meshes and can be integrated into an adaptive process. Anisotropic mesh generators in three dimension are obviously more complex and are usually found only in commercial software. We study our own mesh optimization method so that we instead need software that determines a triangulation of a set of points directly from the metric tensor, without any internal optimization; we use the Simmetrix [33] software package for this purpose. For the mesh optimization process, we use anisotropic centroidal Voronoi tessellations (ACVT) [11,12]. Given a metric tensor, ACVT distributes the nodes by minimizing a cost function so as to improve element quality and reduce sizing distortion. The nodes are the mass centers (centroids) of associated Voronoi regions with respect to the metric determined from a metric tensor. The process of moving the initial nodes to the optimal positions is effected by the generalized Lloyd iteration method. ACVT with an input of the metric tensor returns high-quality meshes to solve convection-dominated problems.

Our study is based on constructing an optimal adaptive strategy that can automatically refine, stretch, and orient the mesh elements so that the approximate solution of (1) can be computed more efficiently and accurately. The rest of the paper is organized as follows. The model problem is introduced and discretized by the SUPG scheme in Section 2. In Section 3, we discuss the metric tensor we use; see[20] for its derivation. Section 4 contains some theoretical and algorithmic discussions about ACVTs. In Section 5, the adaptive algorithm is presented. In Sections 6 and 7, we present the results of computational experiments with known and unknown exact solutions, respectively. Possible extensions are discussed in Section 8.

2 The streamline upwind Petrov-Galerkin method (SUPG)

For a nonnegative integer s, let $H^s(\Omega)$ denote the Sobolev space of functions having square integrable derivatives of order up to s; note that $H^0(\Omega) = L^2(\Omega)$. Let $H_0^1(\Omega) = \{\phi \in H^1(\Omega) : \phi = 0 \text{ on } \Gamma\}$. The standard Galerkin variational formulation of (1) is given by: find $u \in H^1(\Omega)$ such that $u = g$ on Γ and

$$B(u,v) = F(v) \qquad \forall\, v \in H_0^1(\Omega), \tag{2}$$

where

$$\begin{cases} B(u,v) = a(\nabla u, \nabla v) + (\mathbf{v}\cdot\nabla u, v) \\ \quad F(v) = (f,v) \end{cases}$$

with $(u,v) = \int_\Omega uv dxdy$. For convection-dominated problems ($a \ll \|\mathbf{v}\|$), discretizations of (2) using practical grid sizes are not able to capture steep layers without introducing non-physical oscillations. To improve stability, the SUPG method was introduced in [3].

Assume that $\mathcal{T}$ is a triangulation of Ω. Let $V_h \subset H^1(\Omega)$ denote the space of continuous, piecewise linear functions with respect to $\mathcal{T}$. Let $V_{0,h} = V_h \cap H_0^1(\Omega)$ and $g_h \in V_h|_\Gamma$ be a piecewise linear function approximation of g which may be determined by interpolating the given function g. Then, the SUPG variational formulation of problem (1) is given by: find $u_h \in V_h \subset H^1(\Omega)$ such that $u_h = g_h$ on Γ and

$$B_\delta(u_h, v_h) = F_\delta(v_h) \qquad \forall\, v_h \in V_{0,h}, \tag{3}$$

where

$$\begin{cases} B_\delta(u_h,v_h) = B(u_h,v_h) + \sum\limits_{T\in\mathcal{T}} \delta_T(-a\triangle u_h + \mathbf{v}\cdot\nabla u_h, \mathbf{v}\cdot\nabla v_h)_T \\ \quad F_\delta(v_h) = F(v_h) + \sum_{T\in\mathcal{T}} \delta_T(f, \mathbf{v}\cdot\nabla v_h)_T \end{cases}$$

with $\delta_T \in L^\infty(\Omega)$ a non-negative stabilization parameter. Comparing (2) and (3), we see that the SUPG method is a modified version of the standard Galerkin method for which more diffusion is added in the streamline direction to deal with the instability caused by the convective field. Note that (3) is a consistent discretization of (1) since the additional stabilization terms vanish for the exact solution of (1); thus, the SUPG method is referred to as a *consistently stabilized method.*

Many heuristic choices for the parameter δ_T have been proposed; for a review, see [25]. However, finding the optimal choice that does the best job of diminishing non-physical oscillations is still an open problem.

In one dimension and with constant data, the SUPG solution with continuous piecewise linear finite elements on a uniform mesh is nodally exact [7] if

$$\delta_T = \frac{h_T}{2\|\mathbf{v}\|_{L^2(T)}}\Big(\coth(Pe_T) - \frac{1}{Pe_T}\Big),$$

where $Pe_T = \frac{\|\mathbf{v}\|_{L^2(T)} h_T}{2|a|}$ is the mesh Peclet number and h_T is the element length. In [3], it was suggested that the stabilization parameters can be approximated by

$$\delta_T = \frac{h_T}{2\|\mathbf{v}\|_{L^2(T)}} \max\Big\{0, 1 - \frac{1}{Pe_T}\Big\} \tag{4}$$

or

$$\delta_T = \frac{h_T}{2\|\mathbf{v}\|_{L^2(T)}} \min\Big\{1, \frac{Pe_T}{3}\Big\}. \tag{5}$$

For our computational experiments, we choose h_T as the length of the longest edge of the element T projected onto the convective field $\mathbf{v}$.

The formulation (4) of the stabilization parameter means that if the mesh size is not "fine" enough (for $Pe_T > 1$ we have that $h_T > \frac{2|a|}{\|\mathbf{v}\|}$), then extra diffusion is added in the direction of the streamwise direction. So, $\delta_T > 0$ if $Pe_T > 1$, i.e., for the convection-dominated case. Otherwise, the standard Galerkin method is used, i.e., $\delta_T = 0$ in the diffusion-dominated case. The formulation (5) for the stabilization parameter shows that $\delta_T > 0$ for both the convection-dominated and diffusion-dominated cases:

$$\delta_T = \begin{cases} c_1 h_T & \text{if} \quad Pe_T > 3 \\ c_2 \frac{h_T^2}{a} & \text{if} \quad Pe_T \leq 3, \end{cases} \tag{6}$$

where c_1 and c_2 are appropriate positive constants. From our experiments, we have observed that, when the mesh Peclet number is large, i.e., for convection-dominated problems, numerical solutions obtained using the stabilization parameter (5) are more stable.

Using the Lax-Milgram theorem, the existence and uniqueness of the solution of the SUPG variational formulation (3) can be proved because it can be shown that

- the bilinear form $B_\delta(\cdot,\cdot)$ is coercive and continuous with respect to the streamline diffusion norm $\|w\|_{sd} = (a\|\nabla w\|^2 + \delta\|\mathbf{v}\cdot\nabla w\|^2)^{\frac{1}{2}}$, where $\|\cdot\|$ denotes the $L^2(\Omega)$-norm;
- the linear functional $F_\delta(\cdot)$ is continuous with respect to the streamline diffusion norm $\|\cdot\|_{sd}$.

For the standard Galerkin method, $B(v_h, v_h) = a(\nabla v_h, \nabla v_h) + (\mathbf{v}\cdot\nabla v_h, v_h)$ so that the coercivity of $B(\cdot,\cdot)$ becomes compromised as $a \to 0$. For the SUPG method, coercivity does not degrade when $a \to 0$. Similarly, the error bounds also suggest that the standard Galerkin discretization will fail to produce accurate solutions in the convection-dominated case. The error bound for the standard Galerkin method is given by the following result [14].

Theorem 1 *If piecewise-linear approximations are used on a shape regular triangulation of Ω, then, there exists a constant C, asymptotically as $a \to 0$ proportional to the mesh Peclet number $Pe = \frac{\|v\|_\infty h}{a}$, such that*

$$\|\nabla(u - u_h)\| \leq Ch\|u\|_2,$$

where h is the length of the longest element edge and $\|\cdot\|_2$ denotes the $H^2(\Omega)$ norm.

The error bound for the SUPG method is given by the following result [14].

Theorem 2 *If piecewise-linear approximations are used on a uniform mesh with $h > 2a$, then there exists a constant C, bounded independently of a, such that*

$$\|u - u_h\|_{sd} \leq Ch^{\frac{3}{2}}\|u\|_2.$$

The derivation of this bound takes advantage of the fact that the coefficient a is of order h.

$L^2(\Omega)$-norm bounds for the errors for the SUPG method are shown, in [26,31], to be of $O(h^{\frac{3}{2}})$ on general quasi-uniform meshes when linear or bilinear elements are used. The apparent loss of a half-order rate of convergence in the L^2-norm for the SUPG method illustrates the difficulty of solving convection-dominated problems. Because the optimal value of the stabilization parameter is unknown, it might be impossible to achieve the optimal convergence rate, i.e., second-order convergence for L^2-error and first-order convergence for H^1-error, when the solution possesses steep layers. Attempts have been made (see, e.g., [8]) to improve the convergence rate through mesh adaptation.

3 Metric Tensor for Anisotropic Mesh Generation

In the convection-dominated case, (1) has layered solutions that exhibit small variation in some directions but rapid changes in other directions. Therefore, it is natural to use a small mesh size in directions of rapid changes and a larger mesh size in directions of small variations. Consequently, in an anisotropic mesh, stretched elements are created along thin layers of the solution. In this way, it is possible to capture the important features of the solution with a much lower number of anisotropic elements compared with the number of elements for an isotropic mesh. To obtain full control of the shape, size, and orientation of anisotropic elements, a metric tensor is required.

In [19–21], general formulas were developed for metric tensors based on error estimates for polynomial preserving interpolation on simplicial elements. Because the error estimates were redefined in terms of mesh qualities, e.g., geometry, alignment, equi-distribution, and adaptation, the metric tensor reflects the overall quality of the mesh with respect to the approximate solution. We choose a metric tensor given in [21] that, specialized to our needs, can be defined as follows.

Given a function v defined on Ω, let $\mathbb{H}(v)$ denote its Hessian matrix and let $\mathbb{H}(\cdot) = \mathbb{Q}\text{diag}(\lambda_1, \lambda_2)\mathbb{Q}^T$ denote its eigen-decomposition so that $\mathbb{Q}$ is the orthogonal matrix consisting of the eigenvectors and λ_i, $i = 1, 2$, are the eigenvalues of $\mathbb{H}(v)$. Let $\mathbb{H}^+(\cdot) = \mathbb{Q}\,\text{diag}(|\lambda_1|, |\lambda_2|)\,\mathbb{Q}^T$. The adaptation function (this and other terminology used here is adopted from [21]) is defined by

$$\rho = \left\| \mathbb{I} + \frac{1}{\alpha}\mathbb{H}^+(v) \right\|^{\frac{1}{2}} \left(\det(\mathbb{I} + \frac{1}{\alpha}\mathbb{H}^+(v) \right)^{\frac{1}{4}}.$$

The intensity parameter α is defined implicitly through the equation

$$\sigma = \int_\Omega \rho(x)\,dx = \frac{|\Omega|}{1-\beta},$$

where $|\Omega|$ denotes the volume of Ω and β roughly indicates the percentage of mesh points concentrated in the regions of large ρ. In [21], it is recommended that β be chosen in the range 0.5 to 0.8. Then, the metric tensor is given by (see [21, equation (4.10)])

$$\mathbb{M}(\mathbf{x}) = \frac{\rho N}{\sigma} \Big(\det(\mathbb{I} + \frac{1}{\alpha}\mathbb{H}^+(v)) \Big)^{-\frac{1}{2}} \Big(\mathbb{I} + \frac{1}{\alpha}\mathbb{H}^+(v) \Big), \tag{7}$$

where the integer N is the number of "target" elements; see Section 5 for how N is chosen. Note that $\mathbb{M}(\mathbf{x})$ is normalized so that it satisfies the unitary volume condition

$$\int_T \sqrt{\det(\mathbb{M}(\mathbf{x}))}\,d\mathbf{x} = 1 \qquad \forall\, T, \tag{8}$$

where T denotes an element in the mesh.

Due to the regularized form with an intensity parameter α, the metric tensor (7) is a positive definite matrix that can be used to generate an anisotropic mesh satisfying the equi-distribution and alignment conditions. A regularized form of the metric tensor for isotropic meshes is given by [21, equation (4.8)]:

$$\mathbb{M}(\mathbf{x}) = \frac{N}{\sigma}\Big(1 + \frac{1}{\alpha}\|\mathbb{H}(v)\|_p\Big)\mathbb{I}, \tag{9}$$

where $\|\cdot\|_p$ denotes the L^p matrix norm. In our computational experiments, the metric tensors (9) and (7) are used to generate isotropic and anisotropic meshes, respectively. The results of the comparisons agree with the expectation that, in the convection-dominated problem, anisotropic meshes are preferred.

In [4–6], metric tensors similar to that in (7) are discussed; the difference is that the conclude that different exponents may be desirable. It would certainly be valuable to test and compare the effectiveness of metric tensors of the form (7) but with different exponents.

The metric tensor can be given a geometric interpretation. In two-dimensions, the metric tensor is a positive definite 2×2 matrix which we have denoted by $\mathbb{M}$. Let $\mathbf{x}$ be a point in Ω. Then, $\mathbb{M}(\mathbf{x}) = \mathbb{E}^T(\mathbf{x})\mathbb{U}(\mathbf{x})\mathbb{E}(\mathbf{x})$, where

$$\mathbb{E}(\mathbf{x}) = \begin{pmatrix} \cos\theta(\mathbf{x}) & \sin\theta(\mathbf{x}) \\ -\sin\theta(\mathbf{x}) & \cos\theta(\mathbf{x}) \end{pmatrix} \qquad \mathbb{U}(\mathbf{x}) = \begin{pmatrix} \mu_1(\mathbf{x}) & 0 \\ 0 & \mu_2(\mathbf{x}) \end{pmatrix}.$$

This is interpreted to mean that the metric tensor $\mathbb{M}(\mathbf{x})$ has transformed a unit circle (around the point $\mathbf{x}$) into an ellipse. Assuming that the mesh is quasi-uniform under the metric $\mathbb{M}(\mathbf{x})$, the eigenvalues and eigenvectors of $\mathbb{M}(\mathbf{x})$ can be used to determine the element aspect ratio and mesh alignment direction for anisotropic mesh generation. The magnitudes of the axes of the ellipse are given by $1/\sqrt{\mu_1(\mathbf{x})}$ and $1/\sqrt{\mu_2(\mathbf{x})}$. Figure 1 shows an example of how the metric tensor rotates and stretches the coordinate axes.

4 Anisotropic Centroidal Voronoi Tessellations (ACVT)

As discussed above, standard Galerkin finite element methods on a uniform mesh produce inaccurate numerical solutions of singularly perturbed problems. As a result, several different types of nonuniform meshes have been developed. The simplest nonuniform mesh is a piecewise-uniform one which consists of a fine mesh in boundary layer regions, a coarse mesh outside the boundary layer,

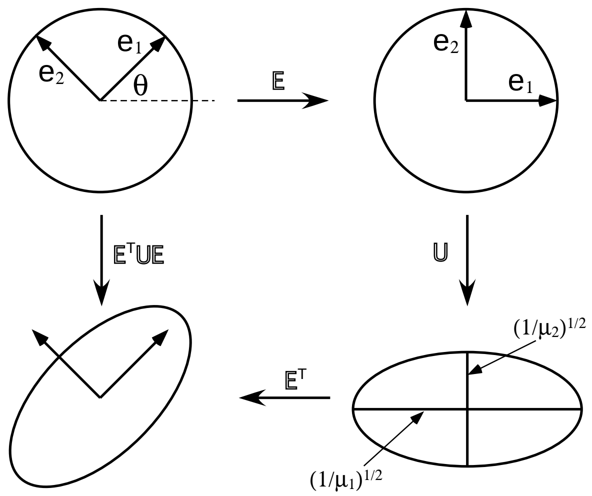


Fig. 1. Geometric meaning of the application of the metric tensor; $\mathbf{e}_1$ and $\mathbf{e}_2$ are the columns of $\mathbb{E}^T$.

and a transition mesh connecting the two. B- and S-meshes (for Bakhvalov and Shishkin) are well-known examples of this type of mesh. Such meshes require knowledge about the structure of the layers. Furthermore, it is essential to properly construct the transition mesh. Due to the above issues, we use the ACVT algorithm [11,27,9,?] as a mesh generator because of its optimality and robustness. The following algorithms and theorems are taken from [11].

4.1 The optimality property

Assume that $\Omega \subset \Re^2$ is a compact set with continuous boundary $\partial\Omega$. Given a metric tensor (a positive definite matrix) $\mathbb{M}(\mathbf{x})$ and a set of distinct points $\{\mathbf{z}_i\}_{i=1}^k$ belonging to $\overline{\Omega}$, the anisotropic Voronoi region (AVR) of a generator $\mathbf{z}_i$ in Ω is defined by

$$V_i(\mathbf{z}_i) = \{\, \mathbf{x} \in \Omega \;\;:\;\; d_{\mathbf{x}}(\mathbf{x}, \mathbf{z}_i) < d_{\mathbf{x}}(\mathbf{x}, \mathbf{z}_j) \quad \text{for } j = 1, \ldots, k,\ j \neq i \,\},$$

where $d_{\mathbf{x}}(\mathbf{x}, \mathbf{p}) = \sqrt{\overrightarrow{\mathbf{xp}}^T\ \mathbb{M}(\mathbf{x})\ \overrightarrow{\mathbf{xp}}}$ is a directional distance and $\overrightarrow{\mathbf{xp}}$ denotes the vector pointing from $\mathbf{p}$ to $\mathbf{x}$. The set $\{V_i\}_{i=1}^k$ is referred to as an anisotropic Voronoi tessellation (AVT) of Ω; the set $\{\mathbf{z}_i\}_{i=1}^k$ is the set of generators of the anisotropic Voronoi tessellation. There are two types of AVR's: interior AVR's for which $\overline{V}_i \cap \partial\Omega = \emptyset$ and boundary AVR's for which $\overline{V}_i \cap \partial\Omega \neq \emptyset$.

Given a metric tensor $\mathbb{M}(\mathbf{x})$, the anisotropic center of mass (centroid) of the

AVR $V(\mathbf{z}_i)$ is given by

$$\mathbf{z}_i^* = \Big(\int_{V(\mathbf{z}_i)} \mathbb{M}\, d\mathbf{x} \Big)^{-1} \int_{V(\mathbf{z}_i)} \mathbb{M}\mathbf{x}\, d\mathbf{x}. \tag{10}$$

Each AVR $V_i(\mathbf{z}_i)$ and its associated generator $\mathbf{z}_i$ define the anisotropic distortion value $F_i(\mathbf{z}_i)$ given by

$$F_i(\mathbf{z}_i) = \int_{V_i(\mathbf{z}_i)} d^2_{\mathbf{x}}(\mathbf{x}, \mathbf{z}_i)\, d\mathbf{x}.$$

The total distortion if given by

$$F(\{\mathbf{z}_i\}_{i=1}^k) = \sum_{i=1}^k F_i(\mathbf{z}_i) = \sum_{i=1}^k \int_{V_i(\mathbf{z}_i)} d^2_{\mathbf{x}}(\mathbf{x}, \mathbf{z}_i)\, d\mathbf{x}. \tag{11}$$

Proposition 1 [11] *A necessary condition for F to be minimized is that, for each $i = 1, \ldots, k$, the generator $\mathbf{z}_i$ of the AVR V_i is itself the centroid $\mathbf{z}_i^*$ of V_i.*

Whenever we have that $\mathbf{z}_i = \mathbf{z}_i^*$ for all i, we refer to the AVT as being a centroidal anisotropic Voronoi tessellation (ACVT). According to Proposition 1, ACVT's can be characterized geometrically by the condition $\mathbf{z}_i = \mathbf{z}_i^*$ for $i = 1, \ldots, k$ or analytically as minimizers of the total distortion function (11).

4.2 Approximation of AVR's and computation of mass centers

To approximate the AVR's of a given set of distinct points $\{\mathbf{z}_i\}_{i=1}^k \subset \overline{\Omega}$ with respect to a given metric tensor $\mathbb{M}$, we use the triangle and neighbor information provided by the anisotropic constrained Delaunay triangulation (ADT) of those points with respect to the given metric tensor. To this end, we use the software package Simmetrix [33]. The outputs are a set of node points $\{\mathbf{z}_i\}_{i=1}^{k'}$, their triangulation $\mathcal{T}$, and the neighbor information for the set of points. For notational purposes, we denote the Simmetrix input-output relation as $(\{\mathbf{z}_i\}_{i=1}^{k'}, \mathcal{T}, \mathcal{N}) = \mathrm{SIM}(\Omega, \mathbb{M}, \{\mathbf{z}_i\}_{i=1}^k)$, where $\mathcal{T}$ denotes the triangulation of the point set $\{\mathbf{z}_i\}_{i=1}^{k'}$ and $\mathcal{N}$ denotes the neighbor information for $\mathcal{T}$, i.e., for each $\mathbf{z}_i$, the points that are connected to it by edges of the triangulation. Note that the number of output points may be different from the number of input points because, depending on the particular properties of the given metric tensor, Simmetrix may change the number of points in order to produce a high-quality ADT.

With the help of the ADT, given an interior vertex $\mathbf{p}$ of the triangulation $\mathcal{T}$, we define the following sets.

- The set $\mathcal{T}_{\mathbf{p}}$ of triangles in the ADT having $\mathbf{p}$ as a common vertex.

- The set $\mathcal{T}^*_{\mathbf{p}}$ of triangles that share the edges of the triangles in $\mathcal{T}_{\mathbf{p}}$ but do not contain $\mathbf{p}$.
- The testing region $\Omega_{\mathbf{p}} = \mathcal{T}_{\mathbf{p}} \cup \mathcal{T}^*_{\mathbf{p}}$ which is the union of the triangles in the ADT that contain the AVR $V_{\mathbf{p}}$ corresponding to $\mathbf{p}$.
- The set of testing vertices $\mathcal{V}_{\mathbf{p}}$ consisting of the vertices in $\Omega_{\mathbf{p}}$, except for $\mathbf{p}$ itself.

Then, the procedure to construct the approximate AVR $\widehat{V}_{\mathbf{p}}$ corresponding to $\mathbf{p}$ is given as follows; see [11] for more details. See Figure 2 for a visual illustration of the discretization used to approximate the AVR.

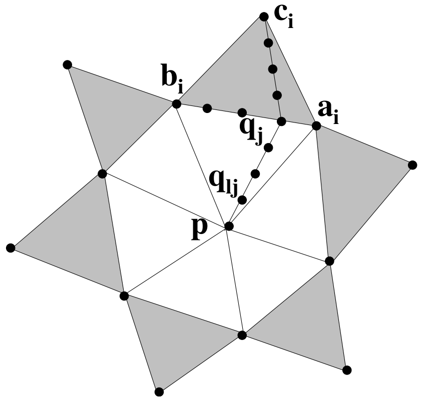


Fig. 2. The discretization used to approximate the AVR in Algorithm 1.

Algorithm 1 *Given a region $\Omega \subset \Re^2$, a metric tensor $\mathbb{M}$, and a set of points $\{\mathbf{z}_i\}_{i=1}^{k}$, determine the ADT $(\{\mathbf{z}_i\}_{i=1}^{k'}, \mathcal{T}, \mathcal{N}) = \mathrm{SIM}(\Omega, \mathbb{M}, \{\mathbf{z}_i\}_{i=1}^{k})$. For each point $\mathbf{p} \in \{\mathbf{z}_i\}_{i=1}^{k'}$, find $\mathcal{T}_{\mathbf{p}}$ and $\mathcal{T}^*_{\mathbf{p}}$ and order their triangles in counter-clockwise (CCW) order. Let $N_{\mathbf{p}}$ denote the number of triangles in $\mathcal{T}_{\mathbf{p}}$. Determine the testing region $\Omega_{\mathbf{p}}$ and the set of testing vertices $\mathcal{V}_{\mathbf{p}}$. Denote by $\widehat{\mathcal{V}}_{\mathbf{p}}$ the set of vertices of the approximate AVR $\widehat{V}_{\mathbf{p}}$. Initially, $\widehat{\mathcal{V}}_{\mathbf{p}} = \emptyset$.*

For $i = 1$ to $N_{\mathbf{p}}$, do the following:

- *Denote the ith triangle of $\mathcal{T}_{\mathbf{p}}$ and $\mathcal{T}^*_{\mathbf{p}}$ by T_i and T_i^*, respectively; near the boundary of Ω, T_i^* might not exist.*
- *Let $\{\mathbf{p}, \mathbf{a}_i, \mathbf{b}_i\}$ denote the CCW-ordered vertices of T_i and $\{\mathbf{b}_i, \mathbf{a}_i, \mathbf{c}_i\}$ the CCW-ordered vertices of T_i^*.*
- *Divide the edge $\mathbf{a}_i\mathbf{b}_i$ into N_e equal sub-edges and order the endpoints of the sub-edges as $\{\mathbf{q}_j\}_{j=0}^{N_e}$ with $\mathbf{q}_0 = \mathbf{a}_i$.*
- *For $j = 0$ to $N_e - 1$, do the following.*
  - · *Divide each of the segments $\mathbf{p}\mathbf{q}_j$ and $\mathbf{q}_j\mathbf{c}_i$ into M_D equal sub-segments.*
  - · *Connect $\mathbf{p}\mathbf{q}_j$ and $\mathbf{c}_i\mathbf{q}_j$, forming the polygonal segment $\mathbf{p}\mathbf{q}_j\mathbf{c}_i$.*
  - · *Order the endpoints of the $2M_D$ sub-segments of the polygonal segment $\mathbf{p}\mathbf{q}_j\mathbf{c}_i$ as $\{\mathbf{q}_{\ell j}\}_{\ell=0}^{2M_D}$ with $\mathbf{q}_{0j} = \mathbf{p}$, $\mathbf{q}_{M_D,j} = \mathbf{q}_j$, and $\mathbf{q}_{2M_D,j} = \mathbf{c}_i$.*
  - · *For $\ell = 1$ to $2M_D - 1$, do the following.*

```
            For $\mathbf{q} \in \mathcal{V}_{\mathbf{p}}$, compare the distances $d_{\mathbf{q}_{\ell j}}(\mathbf{q}_{\ell j}, \mathbf{p})$ and $d_{\mathbf{q}_{\ell j}}(\mathbf{q}_{\ell j}, \mathbf{q})$.
            If there exists $\widehat{\mathbf{q}} \in \mathcal{V}_{\mathbf{p}}$ such that $d_{\mathbf{q}_{\ell j}}(\mathbf{q}_{\ell j}, \mathbf{p}) > d_{\mathbf{q}_{\ell j}}(\mathbf{q}_{\ell j}, \widehat{\mathbf{q}})$, then
                set $\widehat{\mathcal{V}}_{\mathbf{p}} = \widehat{\mathcal{V}}_{\mathbf{p}} \cup \mathbf{q}_{\ell j}$
                exit the $\ell$-index loop and move to the next polygonal segment of
                the $j$-index loop.
            end if
        end for $\{\ell\}$
    end for $\{j\}$

end for $\{i\}$

Connect the points in $\widehat{\mathcal{V}}_{\mathbf{p}}$ which are already in CCW-order to obtain the closed
polygon $\widehat{V}_{\mathbf{p}}$. Then, $\widehat{V}_{\mathbf{p}}$ is regarded as an approximation of the AVR $V_{\mathbf{p}}$ corre-
sponding the point $\mathbf{p} \in \{\mathbf{z}_i\}_{i=1}^{k'}$.
```

In our computational experiments, N_e and M_D are set to 4. Our experiments show that these values provide good accuracy without unnecessarily increasing computation time. Since the AVR $V_{\mathbf{p}}$ is approximated by $\widehat{N_{\mathbf{p}}} = N_{\mathbf{p}} N_e$ triangles of the form $\Delta_i = \Delta \mathbf{p} \mathbf{q}_i^* \mathbf{q}_{i+1}^*, \ \ \mathbf{q}_i^* \in \widehat{\mathcal{V}}_{\mathbf{p}}, \ \ i = 1, \ldots, \widehat{N_{\mathbf{p}}}$, the center of mass (10) of $V_{\mathbf{p}}$ can be approximated using the same triangles:

$$\mathbf{z}_i^* \approx \left(\sum_{i=1}^{\widehat{N_{\mathbf{p}}}} |\Delta_i| \sqrt{\det(\mathbb{M}_i)} \ \mathbb{M}_i \right)^{-1} \sum_{i=1}^{\widehat{N_{\mathbf{p}}}} |\Delta_i| \sqrt{\det(\mathbb{M}_i)} \ \mathbb{M}_i \mathbf{y}_i,$$

where $\mathbf{y}_i$ is the geometric center of Δ_i, $\mathbb{M}_i$ is the metric tensor computed at $\mathbf{y}_i$, and $|\Delta_i|$ is the area of Δ_i.

4.3 An algorithm to construct the ACVT

The generalized Lloyd iteration method is used to produce the approximate ACVT mesh.

Algorithm 2 *Given a compact domain $\Omega \subset \mathbf{R}^2$, a set of points $\{\mathbf{z}_i\}_{i=1}^{k}$ in $\overline{\Omega}$, and a background metric tensor $\mathbb{M}(\mathbf{x})$, construct an initial ADT* $(\mathcal{T}, \{\mathbf{z}_i\}_{i=1}^{k'}) = \mathrm{SIM}(\Omega, \mathbb{M}, \{\mathbf{z}_i\}_{i=1}^{k})$.

1. *Construct the approximate AVR's $\{\widehat{V}_i\}_{i=1}^{k'}$ associated with $\{\mathbf{z}_i\}_{i=1}^{k'}$.*
2. *Compute the centers of mass (centroids) of the approximate AVR's found in step* 1.
3. *Move the points $\{\mathbf{z}_i\}_{i=1}^{k'}$ to the centroid positions.*
4. *Construct an new ADT* $(\mathcal{T}, \{\mathbf{z}_i\}_{i=1}^{k''}) = \mathrm{SIM}(\Omega, \mathbb{M}, \{\mathbf{z}_i\}_{i=1}^{k'})$.
5. *If the new points meet some convergence criterion, terminate; otherwise, set $k' = k''$ and return to step* 1.

5 Adaptive Algorithm

Putting the ingredients defined in Sections 2–4 together, we can define the adaptive anisotropic mesh generation algorithm. We believe that the combination of a stable discretization scheme (the SUPG method) and well-adapted anisotropic meshes (using the metric tensor from [21] and ACVT) can significantly improve the numerical approximation of the convection-dominated problems. Following is our proposed algorithm to determine approximate solutions of (1).

Algorithm 3 *Let $\Omega \subset \Re^2$ be a bounded polygonal domain. Given an initial coarse mesh of Ω, do the following.*

Initial step

Solve (1) *by the SUPG method* (3).
Use the approximate solution to compute the metric tensor (7).

*For each level of the **refinement**, the following steps are done:*

1. *Given the computed metric tensor, the **adaptivity** is done by*
   a) *using Simmetrix to create the initial ADT.*
   b) *constructing the ACVT mesh using Algorithm* 2.
2. *Based on the optimized mesh from Step* 1, *solve the* (1) *by the SUPG method* (3).
3. *Compute the metric tensor* (7) *from the approximate solution from Step* 2.

Note that at the beginning of each level of refinement (step 1a), we use the metric tensor (7) computed from the mesh of the previous level. Based on the metric tensor and the generators of the previous mesh, Simmetrix will automatically add more points to the mesh to satisfy the unitary volume condition (8).

Algorithm 3 determines approximations of the solution of (1) on adaptive anisotropic meshes. To compare this solution with the one on adaptive isotropic meshes, one can replace the anisotropic metric tensor (7) with the isotropic tensor (9).

Since we use piecewise linear finite element approximations, we should note how an approximation to the Hessian matrix in the definition (7) of the metric tensor is determined. Following [21, page 649], solution first derivatives are approximated using a linear least-squares fit to the nodal values of the computed solution and second-order derivatives are obtained in a similar manner, but based on the nodal values of the computed first-order derivatives.

The linear system of algebraic equations resulting from the SUPG method (3) with the stabilization parameter (5) is solved using the iterative method mGMRes with incomplete LU preconditioner. Then, the computed solution is given as an input to generate the metric tensors. The parameter N used in the metric tensors (7) and (9) is prescribed as the number of triangles in the mesh used to determine the approximate solution.

In the next two sections, we present computational examples to illustrate the robustness and efficiency of our adaptive algorithm for solving convection-diffusion problems, including convection-dominated problems. We choose $\beta = 0.5$ for all the examples, except for Example 3 of Section 6.3. There, we compare solutions and meshes for $\beta = 0.5$ with those for $\beta = 0.95$. In order to keep the presentation smooth, we collect the figures and plots in Appendix A.

6 Computational experiments with manufactured solutions

Denote by $N_{\mathcal{T}}$ and $N_{\mathcal{V}}$ the number of triangles and vertices in a mesh, respectively, and let u and u_h denote the exact and approximate solutions, respectively. The convergence rate CR [27] with respect to a norm $\|\cdot\|$ at the refinement level m is roughly computed by

$$CR = \frac{2\log\left(\dfrac{\|e_{h_m}\|}{\|e_{h_{m-1}}\|}\right)}{\log\left(\dfrac{N_{\mathcal{V}_{m-1}}}{N_{\mathcal{V}_m}}\right)}, \tag{12}$$

where h_m denotes the grid size, $N_{\mathcal{V}_m}$ denotes the number of vertices, and $e_{h_m} = u - u_{h_m}$, all at the refinement level m.

6.1 Example 1: regular boundary layers

We consider an example taken from [35]. In (1), set $\Omega = (0,1)^2$ and $\mathbf{v}(x,y) = (1,1)^T$ and choose a from the set $\{10^{-1}, 10^{-3}, 10^{-6}\}$. Choose the exact solution $u(x,y) = xy(1-e^{-\frac{1-x}{a}})(1-e^{-\frac{1-y}{a}})$ that is continuous but has regular boundary layers at $x = 1$ and $y = 1$. The right-hand side and boundary conditions are determined from the exact solution. As observed in Figure A.1, the solution becomes much steeper as $a \to 0$. When $a = 10^{-6}$ and adaptive isotropic meshes are used, the approximate solution contains non-physical oscillations in the layer regions as seen in Figure A.2-left. However, the solution is much smoother on the adaptive anisotropic meshes; see Figure A.2-right. This is due to the fact that the amount of extra diffusion in the SUPG scheme depends on the mesh size h_T inside the layer region. Figure A.3 shows the isotropic and anisotropic adaptive meshes. The anisotropic mesh leads to smaller errors in the L^∞- and H^1- norms for $a = 10^{-1}$ and $a = 10^{-6}$. However, the L^2 error is slightly bigger in the anisotropic case; see Figure A.4. This is due to the averaging properties of the L^2-norm of the errors over the whole domain. So, local phenomena such as oscillations are not captured by this norm. As mentioned in [15], the L^2 error might give misleading information for singularly perturbed problems. This argument is supported by comparing the plots of errors on these two meshes. The convergence rates are optimal or near-optimal for both types of meshes, except for the L^∞ norm when a becomes small. This is expected for the isotropic case since the non-physical oscillations appear in the solution. For the anisotropic case, the H^1-norm convergence rate is still optimal.

6.2 Example 2: interior layer with constant convective field

Choose the exact solution $u(x, y) = \frac{1}{1+e^{-200(\sqrt{x^2+y^2}-0.8)}}$ that is continuous but has an interior layer along the quarter circle $x^2 + y^2 = 0.8^2$. In (1), set $\Omega = (0, 1)^2$, $a = 10^{-8}$, and $\mathbf{v}(x, y) = (2, 3)^T$ and determine the right-hand side and boundary conditions from the exact solution. A similar example was presented in [23], but the equation treated there was the Poisson equations, i.e., without a convective field.

Figure A.5 illustrates the ability of our algorithm to clearly capture the interior layer in this example with both isotropic and anisotropic meshes. Note that the anisotropic mesh with a smaller number of vertices is able to produce a solution with smaller errors than the isotropic mesh, as can be seen from Figure A.7. For both mesh types, the average convergence rates are nearly optimal.

6.3 Example 3: interior layer with variable convective field

This example is taken from [20]. Choose the exact solution $u(x, y) = (1 + e^{\frac{x+y-0.85}{2a}})^{-1}$ that is continuous but has an interior layer along the line $y = -x + 0.85$. Let $\Omega = (0, 1)^2$, $a = 0.005$, and $\mathbf{v} = (u(x, y), u(x, y))^T$, and let the right-hand side and boundary conditions be determined from the exact solution.

Figure A.8 shows the numerical solutions and meshes when $\beta = 0.5$ as specified in the other examples. Note how the mesh elements are stretched along the layers in the bottom of Figure A.9 as compared with the top of Figure A.9.

Similar to Example 2, the anisotropic mesh can achieve smaller errors with fewer vertices compared with the isotropic mesh, as seen in Figure A.10. The convergence rates are nearly optimal for both mesh types.

To compare the isotropic and anisotropic meshes when β is changed from 0.5 to 0.95, refer to Figures A.11 and A.12. Because β indicates roughly the percentage of mesh points concentrated in the layer regions, the larger value of β results in the extremely long thin triangles and a higher point concentration inside the layer. The errors in Figure A.10 when $\beta = 0.95$ are smaller than the ones when $\beta = 0.5$. However, the convergence rate for the L^2 norm shows oscillation for the anisotropic mesh when $\beta = 0.95$. For the isotropic case, it remains optimal. This may be due to an increase in interpolation or discretization errors because of the very thin elements in the anisotropic mesh. To avoid this problem, we can relax the value of β as we observe when we change β from 0.95 to 0.5. This is the reason why $\beta = 0.5$ is used for all the examples although clearly further studies of the effect of the parameter β are called for. Alternatively, it may be possible to place a constraint on the angles of the thin anisotropic elements. Mesh modification such as edge swapping or local smoothing can also be considered in the future.

7 Computational experiments with unknown solutions

In practice, we usually have to deal with more complicated problems which do not have a known solution. Examples 4 to 6, taken from [15], have both components of the convective field $\mathbf{v}$ being negative. This means that, on a unit square domain, the locations of regular boundary layers can happen at the outflow boundaries $x = 0$ and $y = 0$. A Shishkin type mesh is used in [15] and requires this a priori information to construct the mesh. Example 7 is also taken from [15] and has the convective field $\mathbf{v}(x, y) = -(1, 0)^T$. This means that, on a unit square domain, a regular boundary layer can occur at the outflow boundary $x = 0$ while parabolic layers can occur at $y = 0$ and $y = 1$. A corner boundary layer can also occur at a corner of the unit square. The Shishkin mesh type [15] again requires this a priori information for the mesh construction.

7.1 Example 4: regular boundary layers with smooth data

In (1), set $\Omega = (0, 1)^2$ and $\mathbf{v}(x, y) = -(2, 1)^T$ and choose $a = 10^{-1}$ or $a = 10^{-6}$. The right-hand side is given by $f(x, y) = -x^2(1 - x)^2y^2(1 - y)^2$ and the boundary condition is $u(x, y) = 0$ on $\partial\Omega$.

Figures A.13–A.16 show the numerical solutions and meshes for isotropic and anisotropic meshes. Here, we observe similar behavior as in Example 1, where the diffusion term a was also varied. For $a = 10^{-1}$, the isotropic and anisotropic solutions are not distinguishable to the eye. However, non-physical oscillations appear in the isotropic solution when $a = 10^{-6}$ but the solution is still smooth for the anisotropic case. These results support the argument that the adaptive anisotropic mesh is efficient in capturing the layers and avoiding non-physical oscillations for convection-dominated problems. For the remaining examples, we show only numerical solutions on anisotropic meshes.

7.2 Example 5: regular and corner boundary layers with non-smooth data

In (1), set $\Omega = (0,1)^2$, $\mathbf{v}(x,y) = -(2+x^2y, 1+xy)^T$, and $a = 10^{-6}$. The right-hand side is given by $f(x,y) = -(x^2 + y^3 + \cos(x+2y))$ and the boundary condition is given by

$$u(x,0) = 0, \qquad u(x,1) = \begin{cases} 4x(1-x), & x < \frac{1}{2} \\ 1, & x \geq \frac{1}{2}. \end{cases}$$

$$u(0,y) = 0, \qquad u(1,y) = \begin{cases} 8y(1-2y), & y < \frac{1}{4} \\ 1, & y \geq \frac{1}{4}. \end{cases}$$

The boundary data are not differentiable at the points $(0.5, 1)$ and $(1, 0.25)$. Figures A.17 and A.18 show that the regular and corner boundary layers are captured well by adaptive anisotropic mesh refinement.

7.3 Example 6: interior and regular boundary layers

This example is similar to Example 5, except that the boundary conditions are different. Let $\Omega = (0,1)^2$, $\mathbf{v}(x,y) = -(1,4)^T$, and $a = 10^{-6}$. The right-hand side is $f(x,y) = -(x^2 - y^2)$ and the boundary conditions are

$$u(x,0) = 1, \quad u(x,1) = (1-x)^{\frac{1}{6}}, \quad u(0,y) = 1, \quad u(1,y) = \sqrt{1-y}.$$

At the inflow corner $(1,1)$, the exact solution is not differentiable. This incompatibility in the inflow boundary data creates an interior layer in the domain which is a line joining the points $(1,1)$ and $(0.75, 0)$. The layer follows the characteristic line of the convective field. In reference [15], a Shishkin mesh is used with refinement along the outflow boundary only, and was not able to capture the interior layer. The authors of that work observed smearing in the layer area because of the lack of interior mesh refinement. Our method is able to detect this layer automatically and, as a result, the mesh is appropriately refined along the layer. Figures A.19 and A.20 show our results.

7.4 Example 7: parabolic and corner boundary layers

Let $\Omega = (0,1)^2$, $\mathbf{v}(x,y) = -(1,0)^T$, and $a = 10^{-3}$. The right-hand side is $f(x,y) = 0$ and the boundary condition is $u(x,0) = (6\sqrt{3}x(1-x)(2x-1))^3$ and $u(x,y) = 0$ if $(x,y) \in \partial\Omega \backslash \{(x,0)\}$. At $y = 0$, the solution has a parabolic boundary layer and positive values when $x > \frac{1}{2}$ and negative values when $x < \frac{1}{2}$. A corner boundary layer occurs at $(0,0)$. Figures A.21 and A.22 show our results, which are similar to those in [15]. The boundary condition along the line $y = 0$ implies that the solution will be positive for $x > 0.5$ and negative for $x < 0.5$. However, some positive values of the solution can be seen in the neighborhood of the midpoint $(0.5,0)$ for $x < 0.5$. This indicates the influence of the convective flow along the $(-1,0)$ direction.

8 Concluding remarks

Our adaptive anisotropic mesh algorithm has substantially improved the numerical approximation for steady-state convection-diffusion problems. It works well on both diffusion-dominated and convection-dominated problems. The results converges at a quasi-optimal or optimal rate (depending on the characteristics of the layers) and with low computational cost. Due to the efficiency and robustness of our algorithm, non-physical oscillations in the numerical solutions in the layers are not present. Since adaptive anisotropic meshes are used in our algorithm, any local phenomena in the solutions (for example, layers, singularities, etc.) are captured automatically. This capability will allow us to explore more practical and complicated problems in future.

We will continue optimizing our schemes and algorithms to obtain better numerical approximation. Mesh modification techniques such as edge swapping and local smoothing will be explored. These techniques are designed to prevent the elements from getting too thin and causing an increase in matrix conditioning and in interpolation or discretization errors. Comparing the effectiveness of the metric tensor (7) with those discussed in [4–6] is certainly also called for.

Furthermore, implementing the adaptive algorithm in three dimensions is another possibility due to two reasons. First, Simmetrix can generate three-dimensional tetrahedral meshes. Second, the other ingredients (stabilization scheme, metric tensor, and ACVT) can also be generalized to three dimensions. Although the core of the three-dimensional algorithm will be similar to the two-dimensional one, the implementation for three-dimensional mesh generation will likely be quite challenging.

Our studies will also be extended to the time-dependent case. We believe that our achievements in the stationary problem make this task very tractable. We are also interested in constructing the adaptive algorithm for the incompressible Navier-Stokes equation. Understanding its linearized simplified version, i.e., the convection-diffusion equation, gives us more insights on how to apply the SUPG formulation for the Navier-Stokes equations with respect to the mesh adaptation.

9 Acknowledgment

The authors wish to thank Professor Weizhang Huang of the University of Kansas for allowing us to use his code to determine the metric tensor and for several helpful discussions.

References

[1] R. A. Adams; *Sobolev Spaces*, Academic Press, New York, 1975.

[2] L. Angermann; Balanced a posteriori error estimates for finite-volume type discretizations of convection-dominated elliptic problems, *Computing* **55**, 1995, pp. 305–324.

[3] A. Brooks and T. Hughes; Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations, *Comput. Meths. Appl. Mech. Engrg.* **32**, 1982, pp. 199–259.

[4] W. Cao; On the error of linear interpolation and the orientation, aspect ratio, and internal angles of a triangle, em SIAM J. Numer. Anal. **43**, 2005, pp. 19–40.

[5] W. Cao; An interpolation error estimate on anisotropic meshes in R^n and optimal metrics for mesh refinement, *SIAM J. Numer. Anal.* **45**, 2007, pp. 2368–2391.

[6] L. Chen, P. Sun, and J. Xu; Optimal anisotropic meshes for minimizing interpolation errors in L^p-norm, *Math. Comp.* **79** , 2007, pp. 179–204.

[7] I. Christie, D. Griffiths, A. Mitchell, and O. Zienkiewicz; Finite element methods for second order differential equations with significant first derivatives, *Int. J. Numer. Meth. Engrg.* **10**, 1976, pp. 1389–1396.

[8] F. Courty, D. Leservoisier, P.-L. George, and A. Dervieux; Continuous metrics and mesh optimization, *Appl. Numer. Math.* **56**, 2006, pp. 17–145.

[9] Q. Du, V. Faber, and M. Gunzburger; Centroidal Voronoi tessellations: applications and algorithms, *SIAM Review* **41**, 1999, pp. 637–676.

[10] Q. Du and D. Wang; Boundary recovery for three dimensional conforming Delaunay triangulation. *Comput. Meths. Appl. Mech. Engrg.* **193**, 2004, pp. 2547-2563.

[11] Q. Du and D. Wang; Anisotropic centroidal Voronoi tessellations and their applications, *SIAM J. Sci. Comput.* **26**, 2005, pp. 737–761.

[12] Q. Du, Z. Huang, and D. Wang, Mesh and solver co-adaptation in finite element methods for anisotropic problems, *Numerical Methods for PDEs* **21**, 2005, pp. 859-874.

[13] H. Elman and A. Ramage; A characterisation of oscillations in the discrete two-dimensional convection-diffusion equation, *Math. Comp.* **72**, 2003, pp. 263–288.

[14] H. Elman, D. Silvester, and A. Wathen; *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, UK, 2005.

[15] P. Farrell, A. Hegarty, J. Miller, E. O'Riordan, and G. Shishkin; *Robust computational techniques for boundary layers*, Chapman & Hall/CRC Press, 2000.

[16] L. Formaggia and S. Perotto; Anisotropic error estimates for elliptic problems, *Numer. Math.* **94**, 2003, pp. 67–92.

[17] F. Hecht; *Bidimensional Anisotropic Mesh Generator* Technical Report, INRIA, Rocquencourt, 1997. Source code:
`http://www-rocq1.inria.fr/gamma/cdrom/www/bamg/eng.htm.`

[18] I. Heinrich, P. Huyakorn, O. Zienkiewicz, and A. Mitchell; An "upwind" finite element scheme for two-dimensional convective transport equation, *Int. J. Numer. Meth. Engrg.* **11**, 1977, pp. 131–143.

[19] W. Huang; Measuring mesh qualities and application to variational mesh adaptation, *SIAM J. Sci. Comput.* **26**, 2005, pp. 1643–1666.

[20] W. Huang; Metric tensors for anisotropic mesh generation, *J. Comput. Phys.* **204**, 2005, pp. 633–665.

[21] W. Huang; Mathematical principles of anisotropic mesh adaptation, *Communications in Comput. Phys.* **1**, 2006, pp. 276–310.

[22] T. Hughes, L. Franca, and G. Hulbert; A new finite element formulation for computational fluid dynamics VIII. The Galerkin/least-squares method for advective-diffusive equations, *Comput. Meths. Appl. Mech. Engrg.* **73**, 1989, pp. 173–189.

[23] V. John; *A numerical study of a posteriori error estimators for convection-diffusion equations*, *Comput. Meths. Appl. Mech. Engrg.* **190**, 2000, pp. 757–781.

[24] V. John and P. Knobloch; A computational comparison of methods diminishing spurious oscillations in finite element solutions of convection–diffusion equations, *Proc. Conference Programs and Algorithms of Numerical Mathematics* **13** Prague, May, 2006.

[25] V. John and P. Knobloch; *A comparison of spurious oscillations at layers diminishing (SOLD) methods for convection diffusion equations: Part I. Comput. Meth. Appl. Mech. Engrg.* **196**, 2007, pp. 2197–2215.

[26] C. Johnson; *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, 1987.

[27] L. Ju, M. Gunzburger, and W. Zhao; Adaptive finite element methods for elliptic PDEs based on conforming centroidal Voronoi Delaunay triangulations, *SIAM J. Sci. Comput.* **28**, 2006, pp. 2023–2053.

[28] P. Laug and H. Borouchaki; *The BL2D Mesh Generator: Beginner's Guide, User's and Programmer's Manual,* Technical Report, INRIA, Rocquencourt 1996. Source code:
`http://www.iist.unu.edu/∼alumni/software/other/inria/www/bl2d/eng.htm.`

[29] K. Morton; *Numerical Solution of Convection-Diffusion Problems*, Chapman and Hall, London, 1996.

[30] J.-F. Remacle, X. Li, M. Shephard, and J. Flaherty; Anisotropic adaptive simulation of transient flows, *Inter. J. Numer. Meth. Engrg.* **62**, 2005, pp. 899–923.

[31] H.-G. Roos, M. Stynes, and L.Tobiska; *Numerical Methods for Singularly Perturbed Differential Equations – Convection-Diffusion and Flow Problems*, Springer-Verlag, Berlin, 1996.

[32] B. Semper; Numerical crosswind smear in the streamline diffusion method, *Comput. Meth. Appl. Mech. Engrg.* **113**, 1994, pp. 99–108.

[33] Simmetrix software: `http://www.simmetrix.com.`

[34] M.Stynes; *Steady-state convection-diffusion problems*, Acta Numerica, Cambridge University Press, Cambridge, 2005, pp. 445–508.

[35] Z. Zhang; Finite element superconvergence on Shishkin mesh for 2-d convection-diffusion problems, *Math. Comp.* **72**, 2003, pp. 1147–1177.

[36] O. Zienkiewicz and J. Zhu; A simple error estimator and adaptive procedure for practical engineering analysis, *Inter. J. Numer. Meth. Engrg.* **24**, 1987, pp. 337–357.

A Figures and Plots

A.1 Example 1

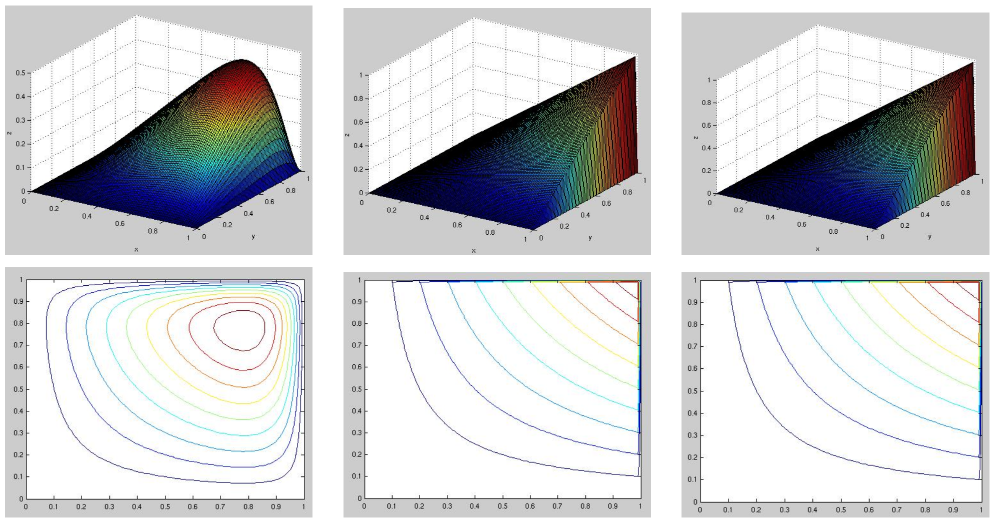


Fig. A.1. Example 1: plots of the exact solution (top) and its contours (bottom) with $a = 10^{-1}, 10^{-3}$, and 10^{-6} (from left to right).

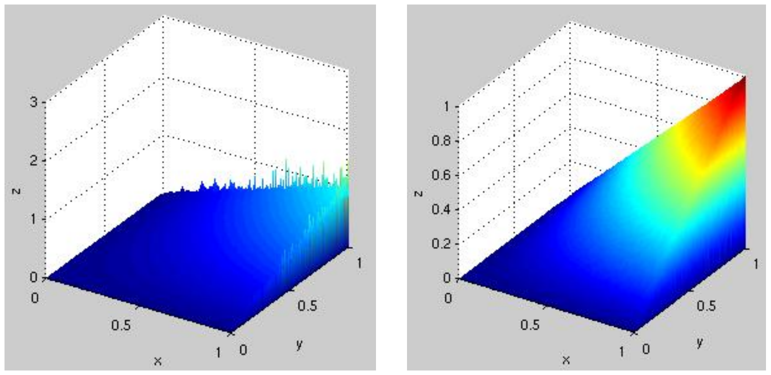


Fig. A.2. Example 1 ($a = 10^{-6}$): approximate solutions on an isotropic mesh with 17, 958 vertices (left) and an anisotropic mesh with 15, 221 vertices (right).

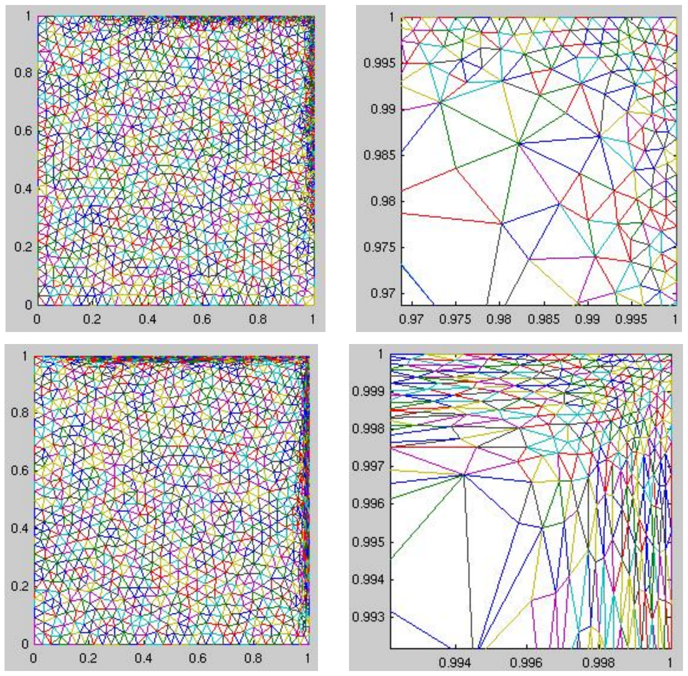


Fig. A.3. Example 1 ($a = 10^{-6}$): the isotropic mesh with $2,273$ vertices (top) and the anisotropic mesh with $2,492$ vertices (bottom) with zoom ins of the upper-right corner on the right.

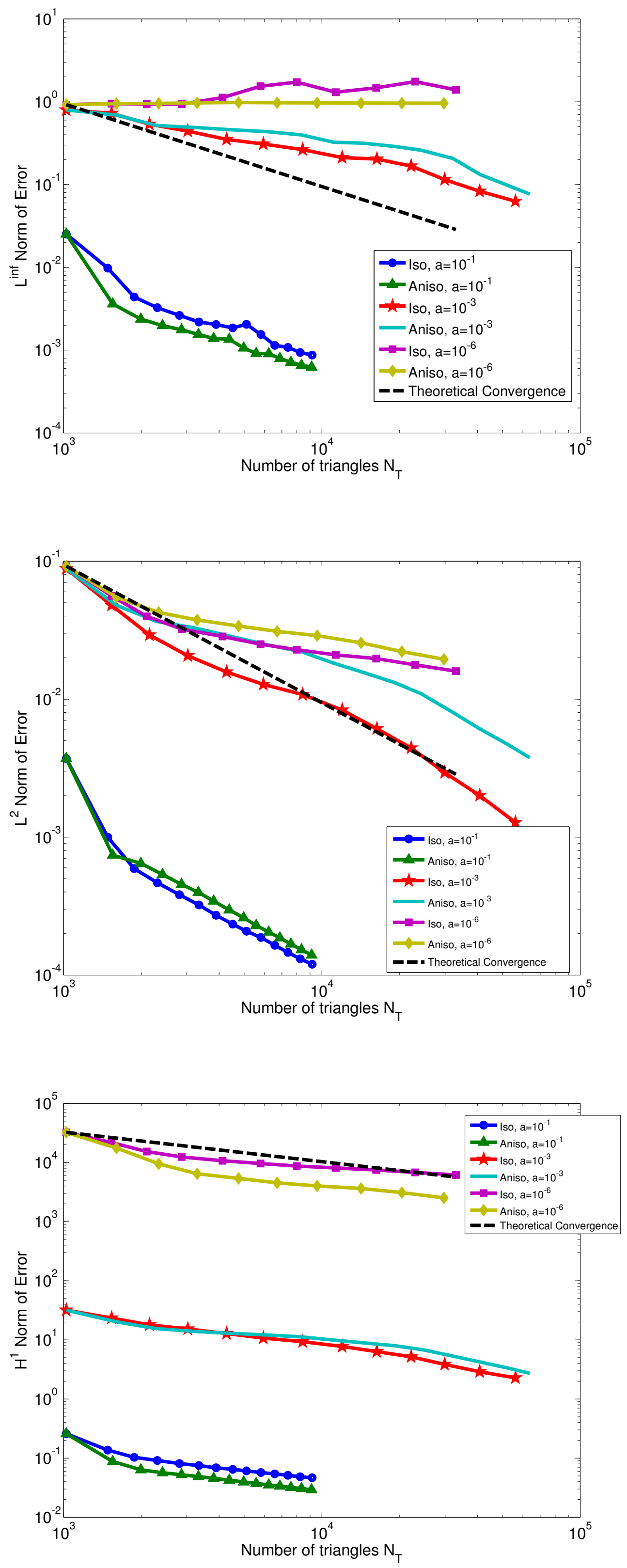


Fig. A.4. Example 1: The L^∞, L^2, and H^1 norms (top to bottom) of the error vs. the number of triangles $N_{\mathcal{T}}$.

A.2 Example 2

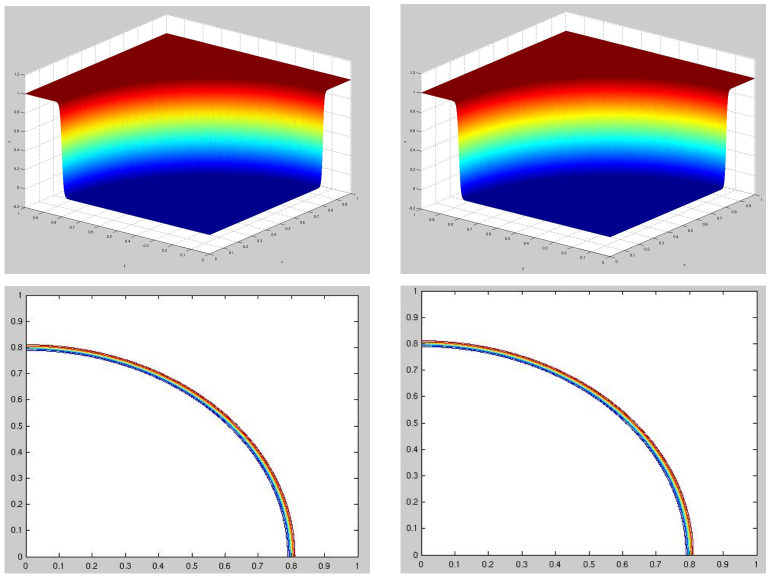


Fig. A.5. Example 2 ($a = 10^{-8}$): plots of the approximate solution (top) and its contours (bottom) on the isotropic mesh with $21,920$ vertices (left) and the anisotropic mesh with $21,840$ vertices (right).

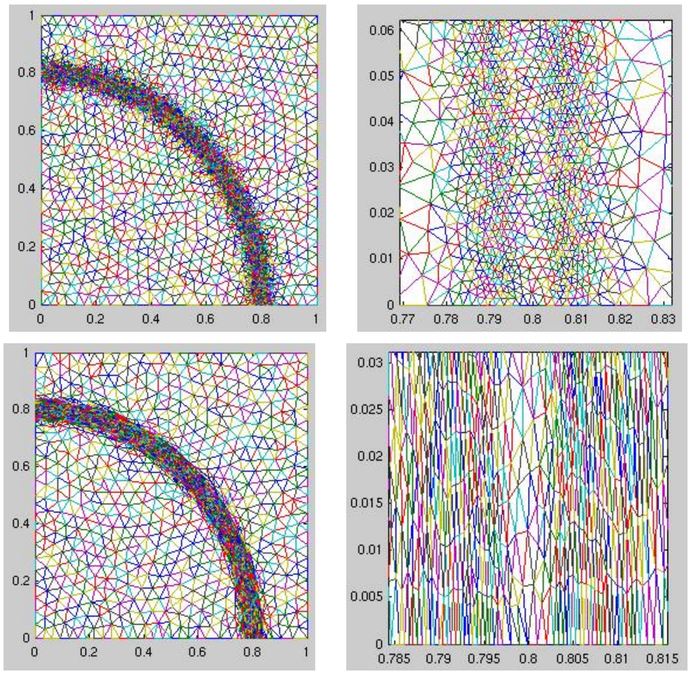
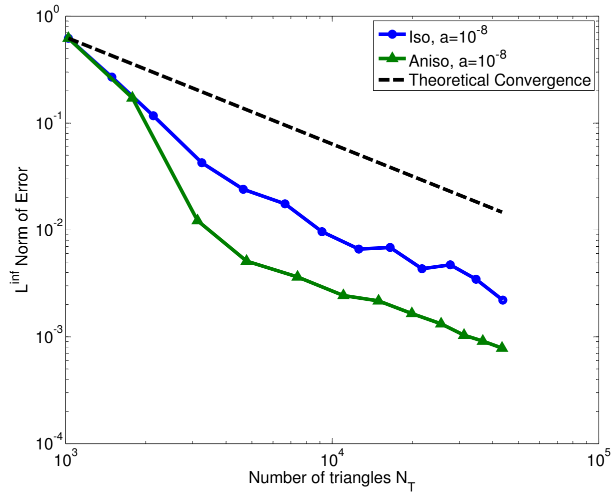
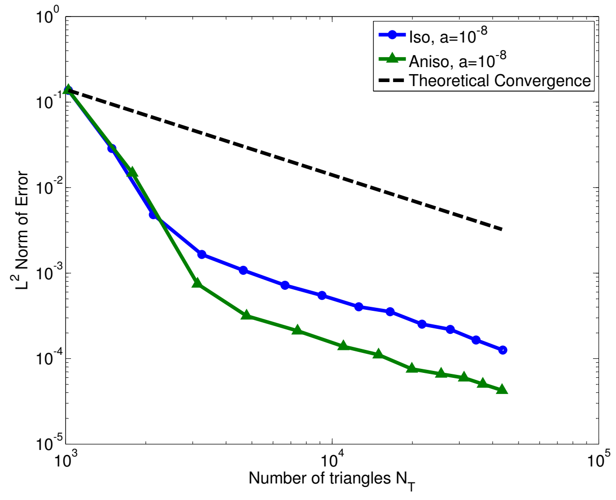


Fig. A.6. Example 2 ($a = 10^{-8}$): the isotropic mesh with $17,404$ vertices (top) and the anisotropic mesh with $15,762$ vertices (bottom) with zoom ins of the interior layer on the right.





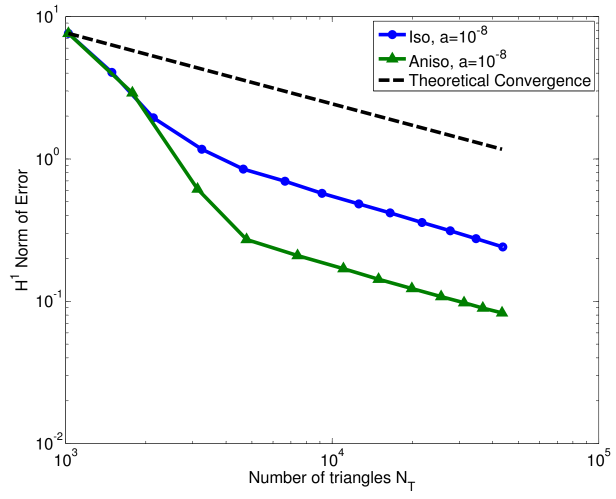


Fig. A.7. Example 2: The L^∞, L^2, and H^1 norms (top to bottom) of the error vs. the number of triangles $N_{\mathcal{T}}$.

A.3 Example 3

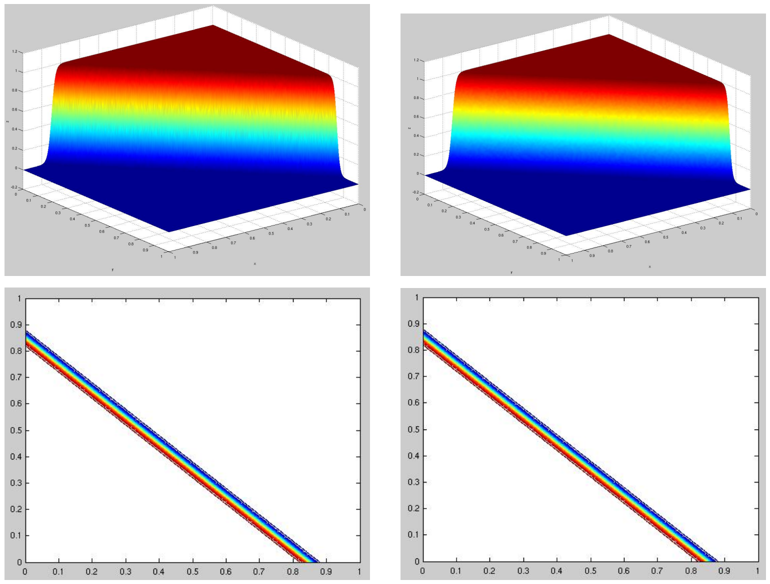


Fig. A.8. Example 3 ($a = 0.005$): plots of the approximate solution (top) and its contours (bottom) on the isotropic mesh with $14,901$ vertices (left) and the anisotropic mesh with $15,315$ vertices (right).

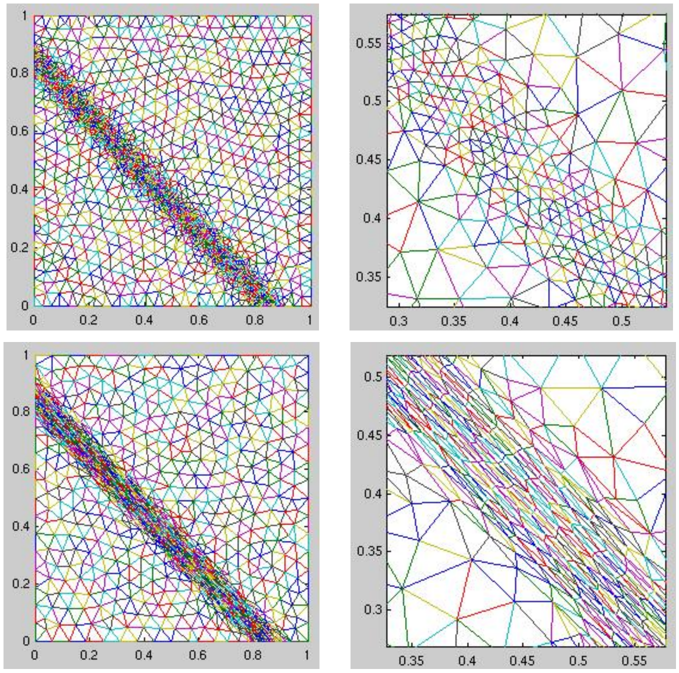
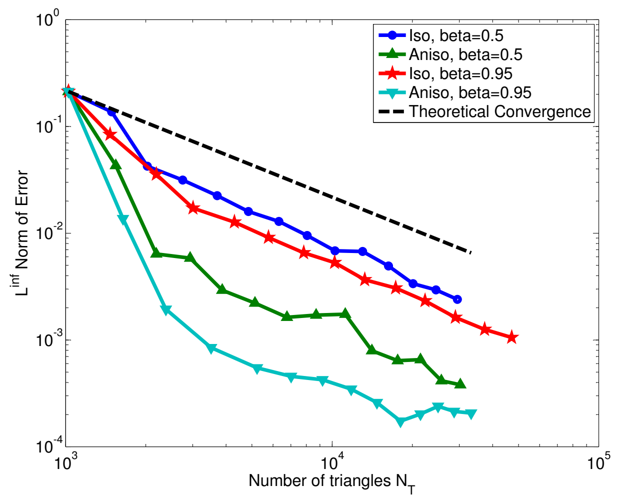
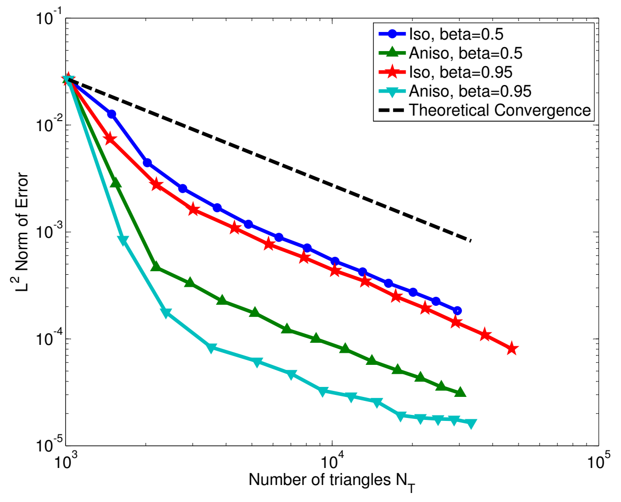


Fig. A.9. Example 3 ($a = 0.005$): the isotropic mesh with $1,444$ vertices (top) and the anisotropic mesh $1,143$ vertices (bottom) with zoom ins of the interior layer on the right.





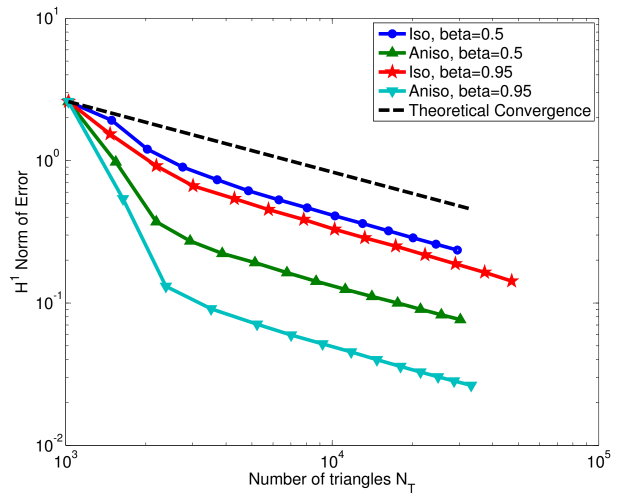


Fig. A.10. Example 3: The L^∞, L^2, and H^1 norms (top to bottom) of the error vs. the number of triangles $N_{\mathcal{T}}$.

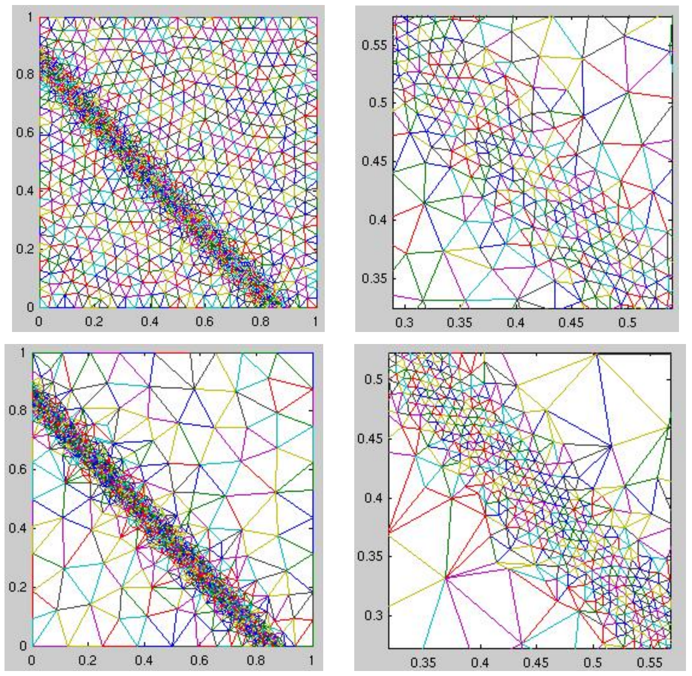


Fig. A.11. Example 3 ($a = 0.005$): isotropic meshes with $\beta = 0.5$ (top, $1,444$ vertices) and $\beta = 0.95$ (bottom, $1,536$ vertices) with zoom ins of the interior layer on the right.

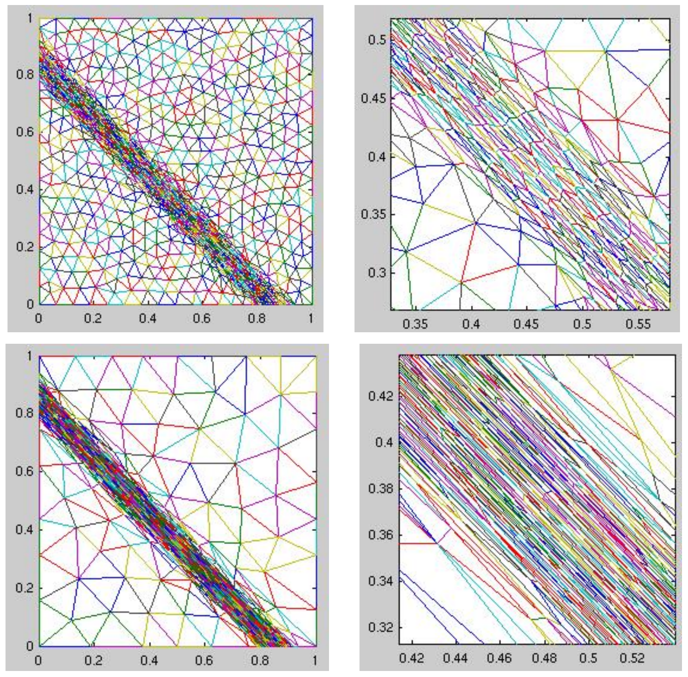


Fig. A.12. Example 3 ($a = 0.005$): anisotropic meshes with $\beta = 0.5$ (top, $1,143$ vertices) and $\beta = 0.95$ (bottom, $1,245$ vertices) with zoom ins of the interior layer on the right.

A.4 Example 4

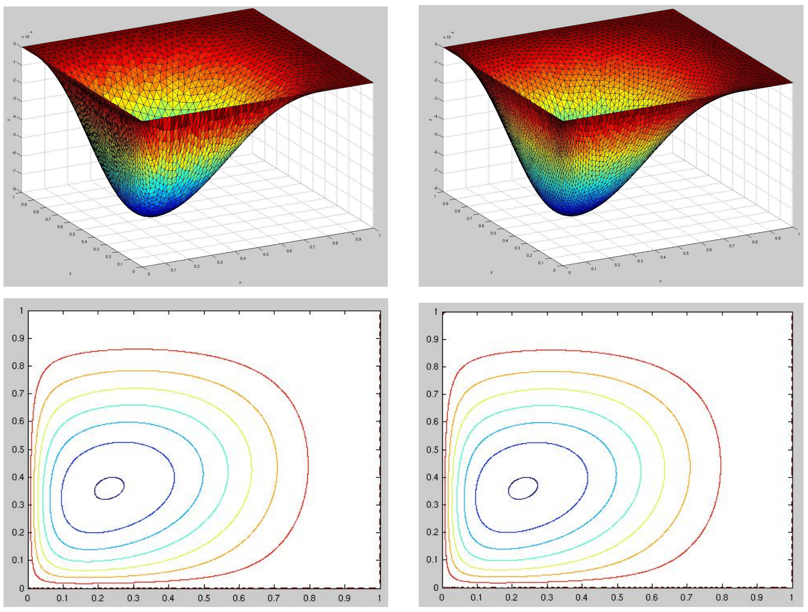


Fig. A.13. Example 4 ($a = 10^{-1}$): plots of the approximate solution (top) and its contours (bottom) on the isotropic mesh with $2,646$ vertices (left) and the anisotropic mesh with $4,065$ vertices (right).

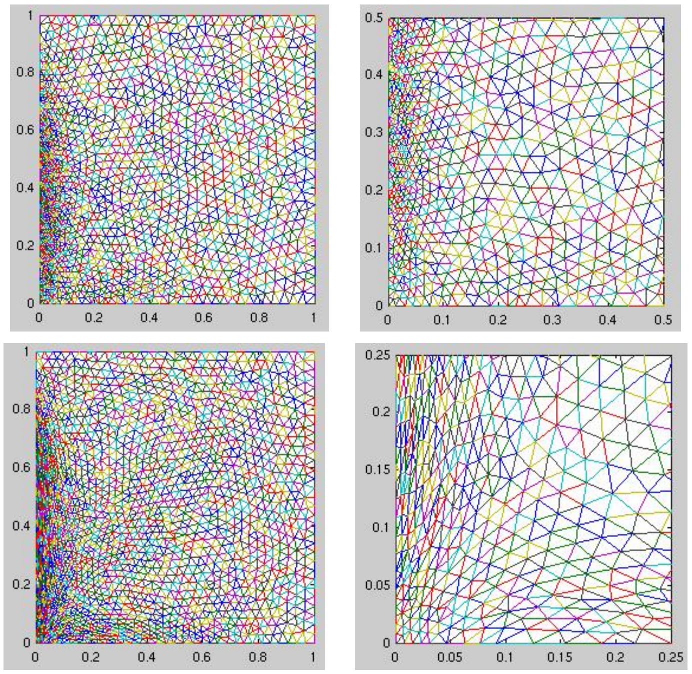


Fig. A.14. Example 4 ($a = 10^{-1}$): the isotropic mesh with $1,414$ vertices (top) and the anisotropic mesh with $1,867$ vertices (bottom) with zoom ins of the lower left-hand corner on the right.

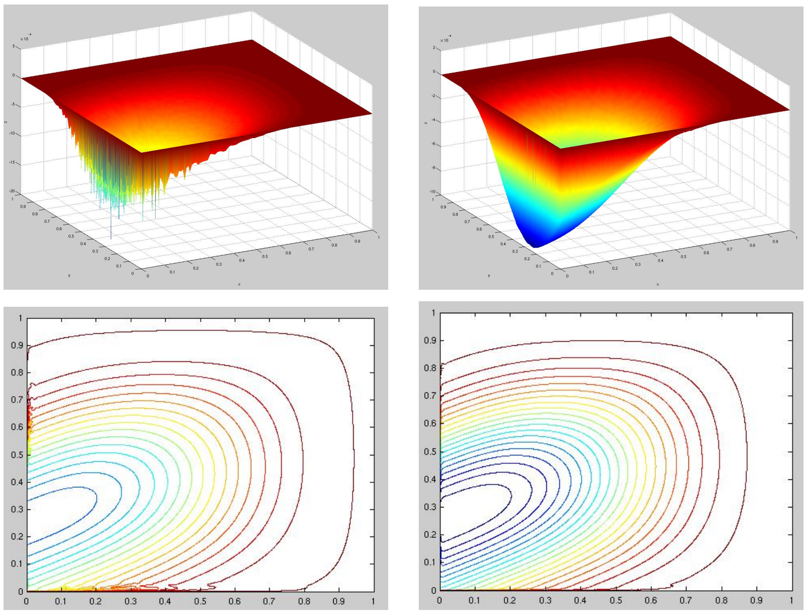


Fig. A.15. Example 4 ($a = 10^{-6}$): plots of the approximate solution (top) and its contours (bottom) on the isotropic mesh with $16,368$ vertices (left) and the anisotropic mesh with $16,423$ vertices (right).

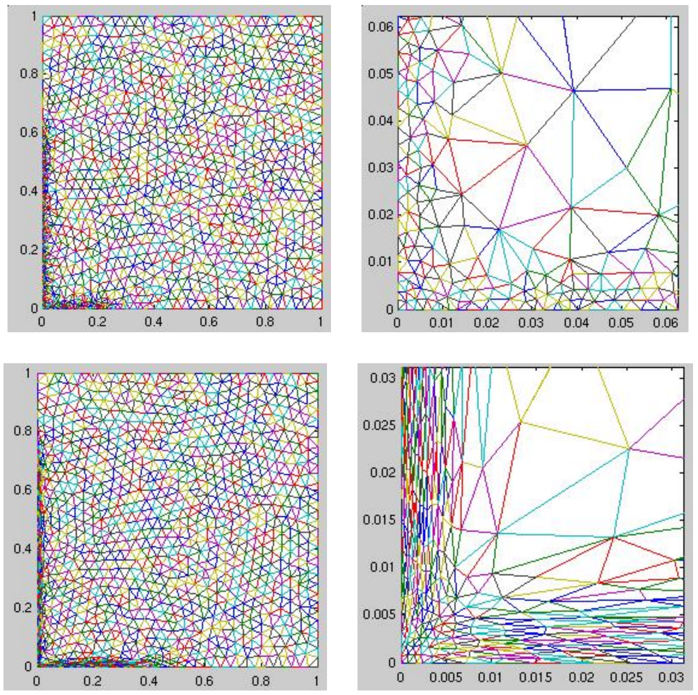


Fig. A.16. Example 4 ($a = 10^{-6}$): the isotropic mesh with $2,064$ vertices (top) and the anisotropic mesh $2,394$ vertices (bottom) with zoom ins of the lower left-hand corner on the right.

A.5 Example 5

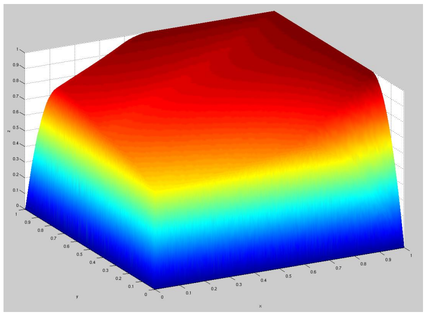
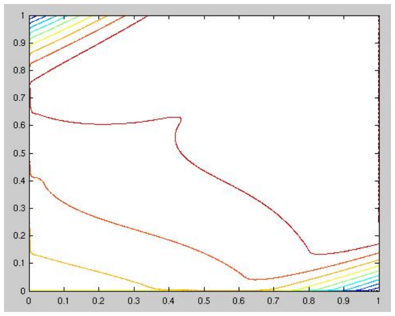



Fig. A.17. Example 5 ($a = 10^{-6}$): plots of the approximate solution (left) and its contours (right) on the anisotropic mesh with $27,635$ vertices.

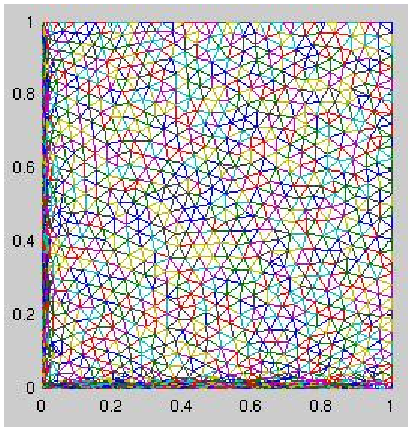
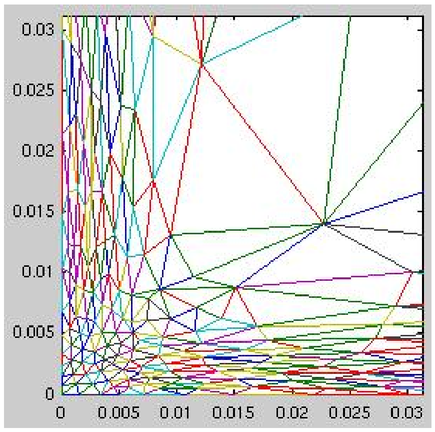



Fig. A.18. Example 5 ($a = 10^{-6}$): the anisotropic mesh with $2,307$ vertices with a zoom in of the lower left-hand corner on the right.

A.6 Example 6

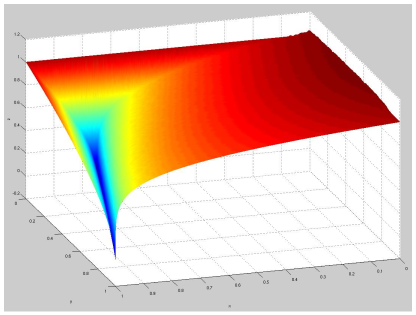
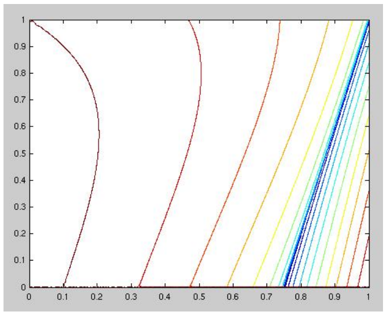



Fig. A.19. Example 6 ($a = 10^{-6}$): plots of the approximate solution (left) and its contours (right) on the anisotropic mesh with $30,350$ vertices.

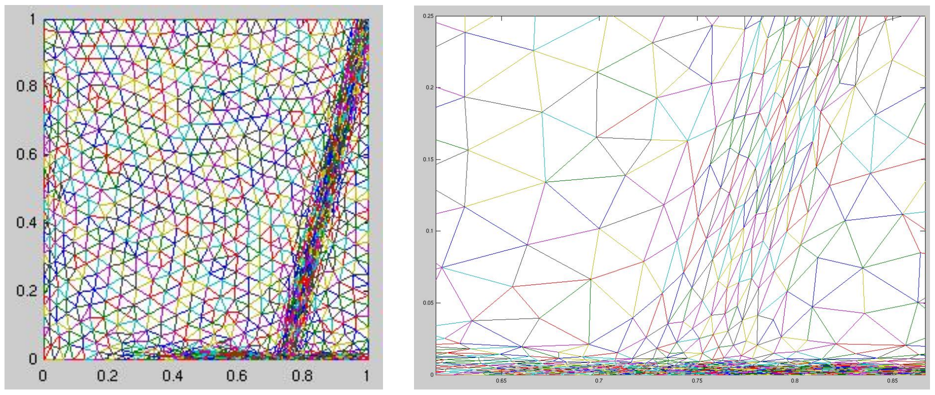


Fig. A.20. Example 6 ($a = 10^{-6}$): the anisotropic mesh with $1,765$ vertices with a zoom in of the layers on the right.

A.7 Example 7

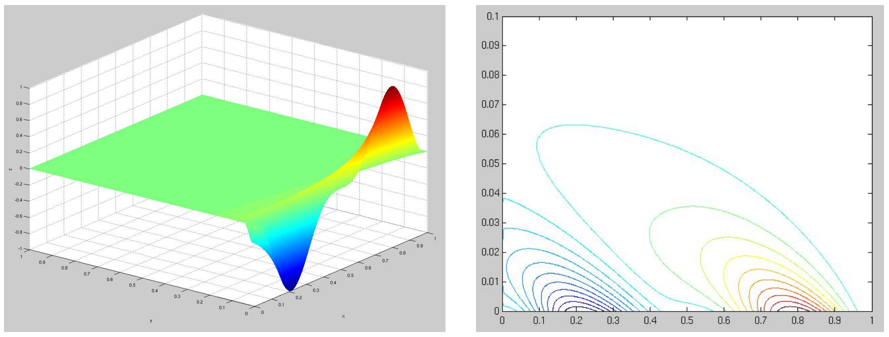


Fig. A.21. Example 7 ($a = 10^{-3}$): plots of the approximate solution and its contours on the anisotropic mesh with $33,006$ vertices.

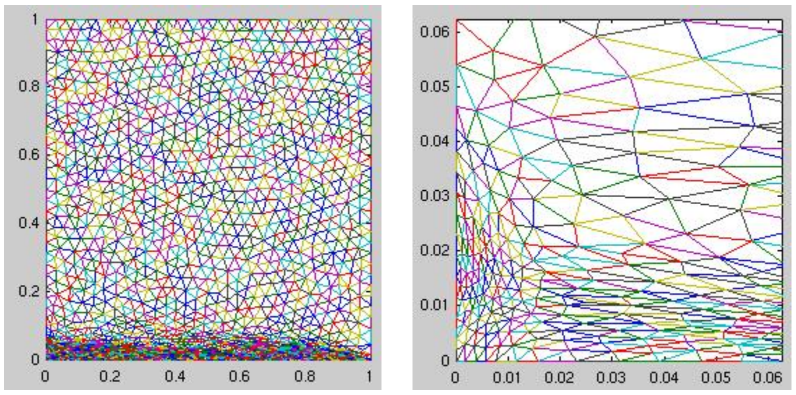


Fig. A.22. Example 7 ($a = 10^{-3}$): the anisotropic mesh with $2,539$ vertices with a zoom in of the layers on the right.