Anisotropic Refinement Algorithms for Finite Elements

Jonathan Goodman
Klas Samuelsson
Anders Szepessy

January 31, 1996

Contents

1 Introduction

Many problems call for *anisotropic refinement*: high resolution is needed in some directions but not in others. Anisotropic refinement fits high aspect ratio elements around high aspect ratio features such as boundary layers[], shocks waves[], shear bands[], and flame fronts[]. It attempts to resolve anisotropic features using a small number of elements. In contrast to the shape regular, or isotropic, refinement problem, anisotropic is poorly understood, difficult, and controversial. In this paper we attempt to construct an anisotropic adaptive refinement algorithm that is modeled on the successful isotropic algorithms [] [].

The shape regularity condition is difficult to abandon. The geometric restrictions it imposes make it possible to design refinement strategies that produce optimal (to within a constant factor) hierarchical adaptive triangulations in many circumstances, optimal within the class of shape regular triangulations. In order to use the greater geometric freedom allowed by irregular, anisotropic refinement, several major difficulties must be overcome, including:

Grid Generation: Find general adaptive algorithms to generate shape irregular triangulations that take optimal advantage of their geometric flexibility.

Error Estimation: Adaptive finite elements are mostly driven by a posteriori error estimates. None of these gives information about directions needing refinement; they do not tell us how to orient high aspect ratio elements.

Accuracy: Ignoring shape conditions completely can result in loss of accuracy or even non convergence. Find ways to avoid this.

Solution Efficiency: Find iterative methods or preconditioners that are as effective on irregular elements as the best methods on regular elements.

Let us discuss these difficulties in more detail, starting with grid generation. Adaptive methods use localization: they locally increase the resolution (as measured by the concentration of elements) in regions that need it. A simple recursive subdivision strategy will yield approximations of optimal order allowed with shape regular elements. We need only to determine which elements need refinement (where greater resolution is needed) and let the code subdivide those elements in an isotropic way. For anisotropic refinement we must choose orientations and aspect ratios for new elements as well as locations. It is a challenge to construct triangulations that take full advantage of this information without adding too many elements. The relationship between shape regular, isotropic refinement and anisotropic refinement is analogous to the relationship between localization and microlocalization in linear partial differential equations[]. Microlocalization localizes both in space and in direction. This is also analogous to the relationship between Young measure and H measure.

We give a heuristic anisotropic refinement algorithm that recursively subdivides elements by adding fewer new vertices than isotropic refinement would add. We have no proof that the algorithm is optimal but we feel that such a proof (or, more likely, an algorithm with a proof) would be very significant since anisotropic refinement is so poorly understood. Even if we can identify directions for refinement, it may be hard to choose triangles that achieve these directions while forming a valid triangulation. See Figure 1.

Violating the shape condition does not automatically result in loss of accuracy. For example, in two dimensions triangulations satisfying a maximum angle condition, $\max|\theta| < \pi - \alpha$, where the maximum is over all angles in the triangulation and $\alpha > 0$, implies convergence,

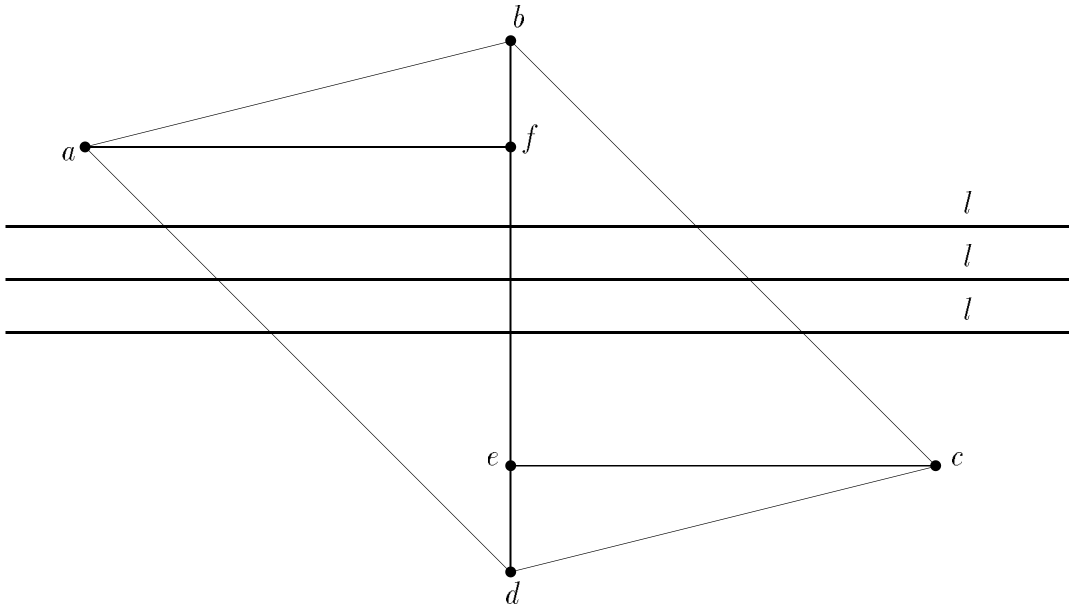

Figure 1: Triangles abc and bcd wish to subdivide to introduce edges aligned with l. Triangle abd creates point f for this purpose while triangle bcd creates point e. Either the two must compromise where to subdivide edge bd or else they must introduce unnecessary edges ae and cf. The complexity increases with the number of triangles. The nonlocality of the refinement is the main difficulty in constructing good anisotropic algorithms.

using finite element methods for standard elliptic problems, see [][]. This condition is satisfied by high aspect ratio, tensor product grids used to resolve boundary layers. Our algorithm does not enforce the maximum angle condition, but we have reason to hope that it works – see below. An anisotropic subdivision algorithm that enforces the maximum angle condition or a proof that our violation of it is safe is very desirable.

The best algorithms for solving finite element equations, such as multigrid, can be ineffective on shape irregular triangulations. Our computations use a multigrid strategy that uses the mesh hierarchy given by the recursive subdivision algorithm. It is reasonable to hope that some of the deep mathematical ideas that work for multigrid with shape regular refinement[] will extend in some form to shape irregular cases.

Our computations examples here include solving the one dimensional problem $u_t + f(u)_x = 0$ by triangulating the (x, t) plane. Fitting a high aspect ratio triangle to the trajectory of a shock in the (x, t) plane can be thought of as a moving mesh method. By triangulating the (x, t) plane with triangles of different sizes, we allow for large time steps in some areas with smaller steps in other areas. An aspect ratio $\triangle t \gg \triangle x$ corresponds to time steps that violate the CFL condition. With space-time triangulations, points are not actually moved so there is no danger of mesh tangling.

This paper discusses several theoretical matters related to general shape irregular refinement. It has been suggested that functional analysis techniques from approximation theory may be helpful for multidimensional triangulations as they have been in one dimension[] and for shape regular refinement in multi dimensions. We argue against this by showing that the "approximation spaces" corresponding to piecewise polynomial approximation on arbitrary triangulations are not even vector spaces: $f+g$ may not be approximable even though f and g are. This is done in Section 2.

An example of shape irregular triangulation leading to non convergence appears in Section 3. We discuss a family of triangulations, $\mathcal{T}_N$ having $h_N \to 0$ as $N \to \infty$ where h_N is the largest of the diameters of the triangles in $\mathcal{T}_N$. The standard Riesz approximation to $\triangle u = f$ with piecewise linear elements on $\mathcal{T}_N$ produces an approximate solution u_N that does not converge to u in any sense as $N \to \infty$.

The important part is Section 4 where we discuss algorithms for recursive anisotropic refinement.

Finite element computations of interacting shock waves and Poisson's equation are presented in Section 5. These computations illustrate the strengths of our approach but also uncover weaknesses that we hope to address in future work.

Section 6 discusses a model approximation problem and proves that anisotropic refinement is better for it. The optimal approximation with N shape regular elements has error $O(1/N)$ while a triangulation without shape regularity with N elements can achieve $O(1/N^2)$. Examples illustrating the behavior of the algorithm in Section 4 for the model approximation problem are given. We compare the convergence rate for refinements based on the exact solution and the finite elements solutions. Finally in Section 7 we draw some conclusions.

2 Approximation theory for general triangulations

There has been much success characterizing functions of one variable that can be approximated at certain rates by free knot splines with N knots (or breakpoints). Here, we show by example that one natural generalization of this to higher dimensions cannot work. As above, if

$$E_N(f) = \min_{\#(T)=N} \min_{f_T \in X_T} \|f - f_T\|_{L^p} \tag{1}$$

then none of the analogues of one dimensional characterizations can hold because "approximation spaces" defined by (1) are not even linear vector spaces: it is possible that f and g are each easy to approximate while $f+g$ is hard. This is stated formally as

Theorem 1 *Given any $\alpha > 0$ and any p with $1 \le p \le \infty$ there are functions f and g with*

$$E_N(f) \le CN^{-\alpha} \quad , \qquad E_N(g) \le CN^{-\alpha} \quad ,$$

but

$$E_N(f+g) \ge CN^{-\alpha/2} \quad .$$

In other words, if $N_\epsilon(f)$ is the smallest number of triangles needed to approximate f to within ϵ, then it is possible that

$$N_\epsilon(f+g) \ge C\left(N_\epsilon(f) + N_\epsilon(g)\right)^2 \quad .$$

The origin of this is simple. The function f is approximated well using N thin triangles that stretch from the bottom to the top of the unit square while g is approximated well using triangles that stretch from the left side to the right. These $2N$ triangles divide the unit square into N^2 regions that are needed to get a comparable approximation to $f+g$.

Our construction for this is based on a self similar function

$$\psi(x)=\sum_{n=0}^{\infty}\lambda^n W(2^n x) \tag{2}$$

where $|\lambda|<1$ and $W(x)$ is the periodic Walsh function defined by

$$W(x+1)=W(x) \quad \text{for all } x,$$

$$W(x)=-1 \quad \text{if} \quad 0\le x<\frac{1}{2}\ ,$$

$$W(x)=1 \quad \text{if} \quad \frac{1}{2}\le x<1\ .$$

Lemma 1 *If $|\lambda|<1$ and $\lambda\neq\frac{1}{2}$, then $\psi(x)$ is not a polynomial on the interval $[0,1)$. In fact, $\psi(x)$ is discontinuous at $x=\frac{1}{2}$ (and at all other dyadic points).*

Proof: As a piece of notation, we write the jump in f as

$$[f](x_0)=f(x_0+0)-f(x_0-0)=\lim_{x\downarrow x_0}f(x)-\lim_{x\uparrow x_0}f(x)\ ,$$

provided that the limits exist. Since the sum (2) converges absolutely and uniformly, we may compute the jump in ψ as the sum of the jumps in the right hand side. The jump in $W(2^n x)$ at $x=\frac{1}{2}$ is

$$W\left(2^n\left(\frac{1}{2}+0\right)\right)-W\left(2^n\left(\frac{1}{2}-0\right)\right)=\begin{cases}2 & \text{if } n=0,\\ -2 & \text{if } n>0,\end{cases}$$

so

$$\begin{aligned}[\psi]\left(\frac{1}{2}\right) &= 2-2\sum_{n=1}^{\infty}\lambda^n\\ &= 2\cdot\left(1-\frac{\lambda}{1-\lambda}\right)\ .\end{aligned}$$

This is different from 0 if $\lambda\neq\frac{1}{2}$ as claimed.

Remark: For $\lambda=\frac{1}{2}$, not only does ψ not have any jumps, but it is linear: $\psi(x)=4\cdot(x-\frac{1}{2})$ for $x\in[0,1)$.

Next, we make precise the sense in which $\psi(x)$ is self similar. Let $I_{k,j}=[j2^{-k},(j+1)2^{-k})$ be a dyadic subinterval. The mapping $x=\xi 2^{-k}+j2^{-k}$ maps the unit interval to $I_{k,j}$. Moreover:

$$\psi(\xi 2^{-k}+j2^{-k})=\lambda^k\psi(\xi)+C_{k,j}\ , \qquad \text{for } \xi\in[0,1). \tag{3}$$

This follows from the observation that for any $n<k$, $W(2^n x)$ is constant on the interval $I_{k,j}$.

The self similarity (3) determines the rate of approximation of ψ by piecewise linear (or piecewise polynomials of any fixed degree). The reason is:

Lemma 2 *Suppose $|\lambda| < 1$ and $\lambda \neq \frac{1}{2}$. If $l(x)$ is any polynomial, then*

$$\int_{I_{k,j}} |\psi(x) - l(x)|^p \, dx \geq C \cdot |I_{k,j}| \cdot \lambda^{pk} \quad , \tag{4}$$

where C depends on λ and the degree of l, but is independent of k, j, and the coefficients of l.

Proof: By the self similarity of ψ, this follows from Lemma 1, and the fact that

$$\min_l \int_0^1 |\psi(x) - l(x)|^p \, dx > C > 0 \quad ,$$

where the minimization is over all polynomials of a fixed degree. To see this, note that the integral is a continuous function of finitely many coefficients and it tends to infinity as any of the coefficients does. Thus the minimum is achieved. The minimum can be zero only if $\psi(x)$ itself is a polynomial, which Lemma 1 says it is not.

To verify (4), define $\tilde{l}(\xi) = \lambda - kl(\xi 2^k + j2^k) + C_{k,j}$, where $C_{k,j}$ is the constant in (3). Then, using (3),

$$\begin{aligned} \int_{k,j} |\psi(x) - l(x)|^p \, dx &= 2^{-k} \lambda^{kp} \int_0^1 \left| \psi(\xi) - \tilde{l}(\xi) \right|^p dx \\ &\geq C \cdot |I_{k,j}| \, \lambda^{kp} \end{aligned}$$

as claimed. This proves Lemma 2.

Given $N = 2^m + 1$ breakpoints, a good approximation to ψ comes from placing break points at the endpoints of the intervals $I_{m,j}$. A piecewise constant approximation to ψ with these breakpoints achieves error

$$\sum_{j=0}^{2^m - 1} \int_{I_{k,j}} |\psi(x) - C_{m,j}|^p \, dx = C \lambda^{mp} \quad .$$

This gives an upper bound on the optimum rate of approximation:

$$E_N(\psi) \leq C \cdot \lambda^m \qquad \text{where } m = \log_2(N). \tag{5}$$

On the other hand, Lemma 2 gives a lower bound of the same order of magnitude. To see this, let $\tilde{\psi}$ be a piecewise polynomial approximation to ψ with N breakpoints and suppose that $N \leq 2^m$, and consider the dyadic intervals, $I_{m+1,j}$, of length $2^{-(m+1)}$. At most half of these intervals contain a breakpoint. In those intervals that do not contain a breakpoint the approximating function is polynomial and Lemma 2 gives it a large error. If M is the number of $I_{m+1,j}$ that do not contain a breakpoint then $M \geq \frac{1}{2} 2^{m+1} = 2^m$, and, from Lemma 2,

$$\begin{aligned} \int_0^1 \left| \psi(x) - \tilde{\psi}(x) \right| dx^p &\geq M \cdot C \cdot \lambda^{(m+1)p} \cdot 2^{-(m+1)} \\ &\geq C \cdot \lambda^{-mp} \quad . \end{aligned}$$

This completes the analysis of the approximation properties of ψ, which are summarized in

Lemma 3 *If $|\lambda| < 1$ and $\lambda \neq \frac{1}{2}$ then*

$$C\lambda^m \leq E_N(\psi) \leq C'\lambda^m \quad , \qquad \textit{where } m = \log_2(N).$$

Here $E_N(\psi)$ is the best approximation to ψ using piecewise polynomials of a fixed degree having N breakpoints. Changing the continuity properties of the approximation at the breakpoints effects only the constants C and C'. The locations of the breakpoints are arbitrary.

We are finally ready to work in two dimensions. The functions f and g in the Theorem are constructed from ψ:

$$f(x,y) = \psi(x) \ , \qquad g(x,y) = \psi(y) \ .$$

The approximation properties of f and g follow from those of ψ and are summarized in:

Lemma 4 *If $|\lambda| < 1$ and $\lambda \neq \frac{1}{2}$, then*

$$C\lambda^m \leq E_N(f) \leq C'\lambda^m \quad , \qquad \textit{where } m = \log_2(N).$$

The same holds for g.

Proof: Let $\tilde{\psi}(x)$ be an approximation to ψ using N breakpoints. Then $\tilde{f}(x,y) = \tilde{\psi}(x)$ is an approximation of f using $2N$ triangles stretching from the bottom to the top of the unit interval. Clearly $\left\|\tilde{f} - f\right\| = \left\|\tilde{\psi} - \psi\right\|$. This, and Lemma 3, proves the upper bound. To prove the lower bound, let $\tilde{f}$ be any piecewise linear approximation to f using N triangles. Then, for almost every y, $\tilde{\phi}_y(x) = \tilde{f}(x,y)$ is a piecewise linear approximation to $\psi(x)$ using at most $3N$ breakpoints (each triangle has 3 edges). From Lemma 3, we have that, for almost every y,

$$\int_0^1 \left|\tilde{f}(x,y) - f(x,y)\right|^p dx = \int_0^1 \left|\tilde{\phi}_y(x) - \psi(x)\right|^p dx \geq C\lambda^{mp} \ ,$$

where, as always, $m = \log_2(N)$. Integrating this over y gives the lower bound that completes the proof of Lemma 4.

The function $h(x,y) = f(x,y) + g(x,y)$ is self similar on dyadic boxes in the way ψ is self similar on dyadic intervals. If $\vec{j} = (j_1, j_2))$, $B_{k,\vec{j}} = I_{k,j_1} \times I_{k,j_2}$, and $B_0 = [0,1) \times [0,1)$ is the unit square, then

$$h(\xi 2^{-k} + j_1 2^{-k}, \eta 2^{-k} + j_2 2^{-k}) = \lambda^k h(\xi,\eta) + C_{k,\vec{j}} \ , \qquad \text{for } (\xi,\eta) \in B_0. \tag{6}$$

This scaling property gives us the analogue of Lemma 2, which is:

Lemma 5 *If $\tilde{h}(x,y)$ is piecewise linear on a triangulation of B_0 having M triangles and a given dyadic subsquare, $B_{k,\vec{j}}$, contains no vertex of any triangle, then*

$$\iint_{B_{k,\vec{j}}} \left|h(x,y) - \tilde{h}(x,y)\right|^p dxdy \geq C \cdot \left|B_{k,\vec{j}}\right| \cdot \lambda^{mp} \ . \tag{7}$$

The constant, C is independent of M.

Proof: (Beginning) The proof is based on scaling, as in the one dimensional case. Using (6), the proof reduces to a statement about approximation on the unit square, B_0. If $\tilde{h}(x,y)$ is piecewise linear on M triangles that form a triangulation of some set in the plane containing B_0 then

$$\iint_{B_0} \left|\tilde{h}(x,y) - h(x,y)\right|^p dxdy \geq C \quad , \tag{8}$$

where C is independent of M but depends only on the absence of vertices inside B_0. The somewhat technical proof of this is deferred until later but the idea is as follows. If there were no positive constant in (8) then there would be a sequence, $\tilde{h}_l$, $l = 1, 2, \ldots$, that approaches h in L^p. Any h that is such a limit is linear on a family of lines that cover the unit square but do not intersect there. Our function h does not have this property since it is discontinuous along the horizontal line $y = \frac{1}{2}$ and along the vertical line $x = \frac{1}{2}$.

The next Lemma completes the proof of the Theorem.

Lemma 6 *If* $|\lambda| < 1$ *and* $\lambda \neq \frac{1}{2}$ *then*

$$C\lambda^{m'} \leq E_N(h) \leq C'\lambda^{m'} \quad , \tag{9}$$

with

$$m' = \frac{1}{2}\log_2(N) \quad . \tag{10}$$

The $\frac{1}{2}$ in (10) is what distinguishes h from f and g, and what distinguishes this from Lemma 4. It says that $h = f + g$ is twice as hard to approximate as f or g. To get the Theorem from here, choose $\lambda = \pm 2^{-\alpha}$ (we must choose $-2^{-\alpha}$ if $\alpha = 1$). Lemma 4 gives $E_N(f) \leq CN^{-\alpha}$ while Lemma 6 gives $E_N(h) \geq C'N^{-\alpha/2}$.

Proof: For the upper bound, there are $(2^m)^2$ dyadic subsquares of the form $B_{m,\vec{j}}$; cutting each across its diagonal gives a triangulation with $N = 2 \cdot 2^{2m}$ elements. If we set $\tilde{h}(x,y) = C_{m,\vec{j}}$ when $(x,y) \in B_{m,\vec{j}}$, where the $C_{k,\vec{j}}$ are from (6), we achieve the upper bound. As for the lower bound, if we take $N = (2^m)^2$ triangles then at least $\frac{1}{4}$ of the dyadic subsquares with side $2^{-(m+1)}$ contain no vertex of a triangle (the number of vertices is not more than $3N$ and the number of such subsquares is $4N$). Arguing as in the one dimensional case gives the lower bound. This proves Lemma 6 and completes the proof of the Theorem.

3 Non convergence with an anisotropic triangulation

In this section we give a simple example showing that the approximation, with an anisotropic mesh of an elliptic problem, can be inconsistent. In this example the orientation of the stretching is exactly $\pi/2$ off from the correct direction. In our adaptive refinement algorithm we seek to find the correct direction and hence try to avoid the trouble in the following example.

Consider the x-independent function $w(x,y) \equiv y^2/2$, $(x,y) \in \boldsymbol{R}^2$, which solves

$$\Delta w = 1, \quad (x,y) \in \boldsymbol{R}^2. \tag{11}$$

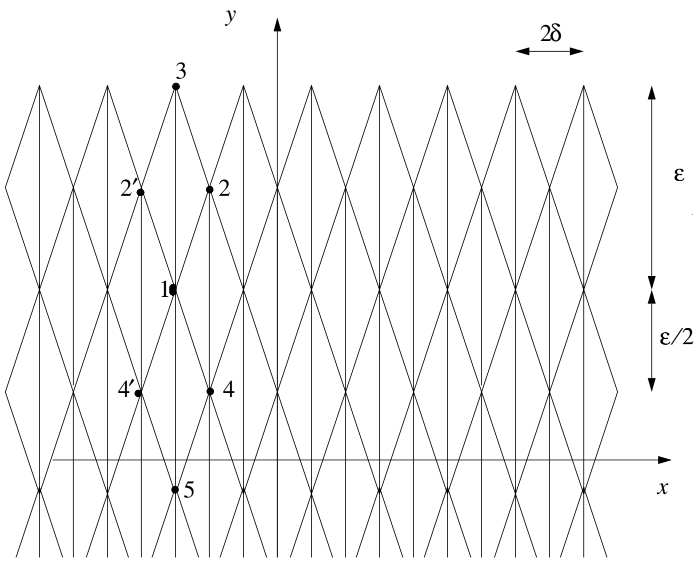

Figure 2: Triangulation for the non convergent example (11).

Let the triangulation of $\boldsymbol{R}^2$ consist of triangles, with base ϵ and hight δ, oriented as in Fig. 2. Discretize (11) by the finite element method with piecewise linear elements, see (16) in Section 4.3. Then, the corresponding matrix problem

$$A\tilde{w} = b, \tag{12}$$

is easy to obtain; by the symmetry in the mesh and the right hand side, each equation takes the same form. That is, the stiffness matrix is the same for all nodes. Moreover, since the problem is independent of the x-coordinate, the approximate solution $\tilde{w}$ is the same in the nodes 2 and $2'$ and in the nodes 4 and $4'$, respectively, in Fig. 2. In the limit when $\delta \ll \epsilon$, the equation in each node is, to leading order δ/ϵ,

$$\frac{\epsilon}{\delta}(\frac{\tilde{w}_5}{4} - \tilde{w}_4 + \frac{\tilde{w}_1}{2} - \tilde{w}_2 + \frac{\tilde{w}_3}{4}) = 3\epsilon\delta.$$

Therefore, each equation in (12) is a stable and a consistent approximation of

$$\frac{d^2u}{dy^2} = 3(\frac{\delta}{\epsilon})^2,$$

to leading order in δ/ϵ. In the limit when $\delta \to 0$, for fixed ϵ, the finite element equation (12) approximates the equation $u'' = 0$, instead of the correct $u'' = 1$.

4 Algorithms for anisotropic refinement

In this section we describe how repeated subdivision of triangles in a mesh can be used to generate an anisotropic mesh, that is, a mesh that can contain triangles with large aspect ratios.

We first describe the basic structure of an *ADAPTIVE ALGORITHM*:

0. Let an initial coarse triangulation $\mathcal{T}_0$ be given, set $k = 0$.

1. Solve the discrete problem on mesh $\mathcal{T}_k$.

2. Compute for each element of $\mathcal{T}_k$ an estimate of the error.

3. If the estimated error is small in all elements **stop**. *Otherwise, construct next mesh $\mathcal{T}_{k+1}$ by refining elements with large errors. Increment k and go to 1.*

Our implementation of steps 1, 2 and 3 is based on

1. Discretization by the streamline diffusion finite element method and solution of the discrete equations by the multigrid method.

2. Refinement criteria that estimates the interpolation / approximation error of the approximate solution in anisotropic elements.

3. Automatic anisotropic mesh generation by mesh enrichment.

The emphasis of the proposed algorithm for anisotropic mesh refinement will be on efficiency and robustness. We compare these aspects of the algorithm with an isotropic mesh enrichment algorithm. To give examples of the behavior of the proposed anisotropic algorithm, we will study two model problems. The first problem is Burgers' equation discretized in a space-time region, where the initial condition yields a solution with shocks. The second model problem is Poisson's equation where data is chosen such that the solution has anisotropy. For both these model problems the exact solutions are known and hence we can compare the convergence rate for the sequence of solutions generated by the isotropic and anisotropic refinement algorithm, respectively. For the elliptic problems this comparison is done for both piecewise linear and quadratic finite elements.

4.1 The mesh enrichment algorithm

We will remove the restriction that elements of a mesh have to be shape regular. In addition to decide the location where the mesh needs to be refined, it is necessary to determine aspect ratio and orientation of the elements. The orientation of the mesh in our algorithm will be based on estimates of the interpolation errors $||u - \pi u||_{L^\infty(K)}$ or $||\nabla(u - \pi u)||_{L^2(K)}$, where u is the exact solution and π is the usual nodal interpolant onto piecewise linear or quadratic finite elements on a triangle K. The interpolation error is for piecewise linear elements a certain function depending on second derivatives of u, and for piecewise quadratic elements on third order derivatives of u.

The exact solution is of course unknown, therefore we try to estimate second and third order derivatives based on the approximate solution. For elements with linear basis functions we estimate the Hessian $H(u) = (u_{xx}, u_{xy}; u_{yx}, u_{yy})$ by using the approximate solution, and for quadratic elements, we need third order derivatives. Section 4.5 describes this computation of higher order derivatives. The details of the refinement algorithm will differ somewhat between our model problems. We start by describing the principal structure of the mesh enrichment algorithm.

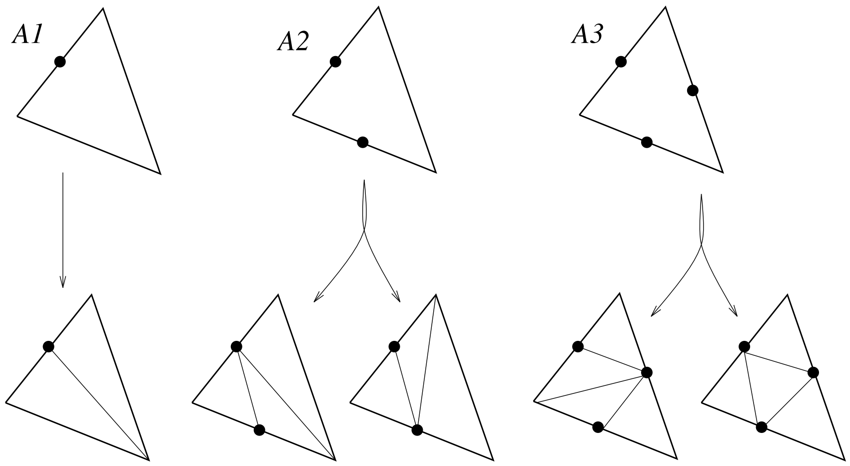

Figure 3: Refinement for anisotropic refinement. For the cases of two or three added nodes there are alternatives in the internal refinement of a triangle.

Let a triangulation and the corresponding approximate solution be given. Our algorithm is based on hierarchical refinement of the triangulation with the following two restrictions: 1) the constructed sequence of meshes will be nested; 2) nodes can only be added at midpoints of edges.

Anisotropic triangles are created recursively by adding new nodes at some midpoints of edges of a triangle and then dividing the triangle into two, three or four triangles, see Fig. 3. This process can be described by the following *BASIC MESH ENRICHMENT ALGORITHM*:

i) Decide for each triangle if it should be refined.

ii) Choose edges of the triangle where a new node shall be added.

iii) Subdivide triangles while choosing between certain alternatives, illustrated in Fig. 3.

An isotropic refinement algorithm consists mainly of step i). The steps ii) and iii) are in this case straightforward. In our anisotropic algorithms steps i) and ii) are closely related.

We illustrate steps i)-iii) for the anisotropic refinement algorithm by briefly describing one possible way to determine the edges where new nodes should be added. This will with minor changes be the algorithm used for the Burgers model problem. To start, the refinement is motivated by

Lemma 7 *Let f be a quadratic polynomial on a triangle K, and let π $C^0(K) \to P_1(K)$ be the nodal interpolant. Then,*

$$\max_{i=1,2,3} E_i \leq \max_{x\in K} |f - \pi f| \leq \frac{4}{3} \max_{i=1,2,3} E_i \ , \tag{13}$$

where $E_i \equiv |\vec{a}_i^t f'' \vec{a}_i|/8$ is $|f - \pi f|$ evaluated at midpoints of edges $i = 1,2,3$. Here the vectors $\vec{a}_i$ are defined as the differences between the end points of the edges $i = 1,2,3$, f'' denotes the Hessian of f and t is the transpose.

Proof: We represent the polynomial $f - \pi f$ by the standard six quadratic basis functions and since $f - \pi f$ is zero at the vertices of K, we obtain

$$f - \pi f = c_4\psi_4 + c_5\psi_5 + c_6\psi_6, \tag{14}$$

where

$$\psi_4 = 4\phi_2\phi_3, \quad \psi_5 = 4\phi_1\phi_3, \quad \psi_6 = 4\phi_1\phi_2.$$

The ϕ_i, $i = 1, 2, 3$ are the linear basis functions, which are one at node i and zero at the other nodes and c_4, c_5, c_6 are coefficients. Evaluation of $f - \pi f$ at the edge midpoints give that $c_4 = E_1$, $c_5 = E_2$ and $c_6 = E_3$, which proves the lower bound. For $x \in K$ the basis functions ψ_4, ψ_5 and ψ_6 are positive, and hence to prove the upper bound it suffices to consider the case $E_1 = E_2 = E_3 = \max_i E_i$ to maximize $|f - \pi f|$. It is easy to see that the maximum of

$$|f - \pi f| = 4 \max_i E_i(\phi_2\phi_3 + \phi_1\phi_3 + \phi_1\phi_2),$$

is attained at the center of gravity of K, where it takes the value $4/3 \max_i E_i$. //

For each edge in the mesh we compute the quantity

$$E = |\vec{a}^t H \vec{a}|/8, \tag{15}$$

where $\vec{a}$ is the vector formed by the edge, and H is a local estimate of the Hessian of the solution. Let all edges with E larger than a given tolerance TOL be marked for refinement. Each marked edge will be split into two, by addition of a new node at the midpoint. To keep track of the chosen refinement, each element use a so called enumeration variable R_t which take values from the set

$$\{UNKNOWN, UNREFINED, ISOTROPIC, A1, A2, A3, P1, P2, A1P1\}.$$

The default value of R_t is $UNKNOWN$. Every triangle of the mesh is examined and if found not to need refinement, the variable R_t is set to $UNREFINED$. For a regular refinement of a triangle into four (similar) triangles, we set R_t to $ISOTROPIC$. For anisotropic refinement where one, two or three edges are to be refined, R_t is set to $A1$, $A2$ or $A3$, respectively. Finally, the last three enumerations, $P1$, $P2$ and $A1P1$, are used for the case that refinement of neighbor triangles imply refinement of the triangle. Here $P1$ and $P2$ stands for that one or two edges have been marked from neighbor triangles. This concludes step ii) where all triangles of the mesh has been examined.

We continue with step iii). When R_t takes the value $A1$ or $P1$ it is clear how to refine: there exists only one way to subdivide the triangle. However, for the $A2, A1P1, P2$ and $A3$ cases we have refinement alternatives, see Fig. 3, where we can choose how the internal edges in the triangle should be placed. Since $E_i = |\vec{a}_i^t H \vec{a}_i|/8$ estimate the size of errors at midpoints of edges, it is natural to choose the refinement alternative which will give the lowest value of E_i in the refined mesh, assuming that the Hessian $H(u_h)$ will not change much. In short, our anisotropic refinement algorithm based on E_i can be stated:

Divide "long" edges and add "short" edges, where long and short is measured by the size of the quantities E_i.

We will now describe the *isotropic refinement algorithm* to which the anisotropic algorithm will be compared. The same error indicator is used as in the anisotropic case, but this time all edges of the triangle are marked, when at least one of the edges indicate that the error is above the given tolerance. Special treatment is used for triangles with hanging nodes, i.e., triangles where one or two edges were marked in the examination of the neighbor triangles. The triangles with hanging nodes are refined as they are if the longest edge is among the marked edges, otherwise the longest edge is marked for bisection. But by bisecting the longest edge the neighbor gets a hanging node, and therefore needs to be checked in the same way. By recursion, the process continues until terminated by that the edge is a boundary edge or that the element on "the other side" has a marked longest edge. It is clear that the length of these marked edges are monotonically increasing, and therefore the "hanging node" algorithm will terminate in finitely many steps.

After this step, each element is individually refined. A triangle with three marked edges is refined isotropically into four similar triangles. If two edges have been marked, then the alternative for which the internal added edge has least Euclidean length is chosen (this can be seen as the anisotropic case with Hessian $H = (1, 0; 0, 1)$).

Remark: If the initial triangulation consists of triangles with angles $\pi/2, \pi/4$ and $\pi/4$, then all triangles of the isotropically refined meshes will be of that shape. For all refined triangles the longest edge is bisected, which ensures that the minimal angle in triangulations refined by the above isotropic refinement will be at least half of the minimal angle in the original triangulation.

4.2 Program structure

The refinement algorithm described in Section 4.1 is implemented in the C++ programming language. The main advantage with C++ compared to FORTRAN is that the mesh refinement part is easier to program and maintain.

The basic C++ classes used in the mesh generation are *point, edge* and *triangle*. The data in the class *point* consists of two coordinates (x, y). The class *edge* consists of pointers to the endpoints of an edge and pointers to the two triangles sharing that edge. An object of the class *triangle* has three pointers to its edges. Each of the objects above, *point, edge* and *triangle*, are given a unique label number which is used to address arrays, e.g., assume that t is an object of the class *triangle*, then an array can be indexed by t.

The classes *edge* and *triangle* are also part of a tree structure consisting of pointers to their parents and descendents. This hierarchy of meshes is used in the multigrid solver. A neighbor to a triangle on the finest mesh is found from its edges which has pointers to the two triangles that share that edge. These pointers are only correct for the finest level, because an edge can be a part of more than one mesh. The correct neighbor triangle on a given coarser level is found by going to the triangle on "the other side" of the edge and in the tree structure seek its parent triangles until the triangle on the same given level is found.

In the refinement procedure of a mesh, objects of the classes *rtriangle* and *redge* are temporarily used. The class *rtriangle* contains the enumeration variable R_t with information about the edges to be marked for refinement and how to refine triangles, cf. Section 4.1.

Finally, the class *redge* is used to construct new nodes and subedges.

4.3 Finite element discretization and multigrid

For Poisson's equation we use the standard Galerkin method: Find $u_h \in V_h$ such that

$$(\nabla u_h, \nabla v) - (f, v) = 0 \quad \text{for all} \quad v \in V_h. \tag{16}$$

For Burgers' equation $u_t + uu_x = \epsilon u_{xx}$, $u(\cdot, 0) = u_0$, we use the streamline diffusion stabilization: Find $u_h \in V_h$ such that

$$(u_{h,t} + u_h u_{h,x}, v + \delta(v_t + u_h v_x)) + \epsilon(u_{h,x}, v_x) + \hat{\epsilon}(\nabla u_h, \nabla v) = 0 \quad \text{for all} \quad v \in V_h. \tag{17}$$

Let us describe the notation. Here V_h is the standard space of continuous, piecewise polynomial functions. A triangle K has two length scales. We define the large length scale $H = \text{diam}(K)$ and the small h by the relation $hH = \text{area}(K)/2$; in other words: h is the height of the triangle with the longest edge, of length H, as base. The aspect ratio of a triangle is defined as H/h. The small length scale is used in the computation of the streamline diffusion parameter $\delta = h/2$ and the shock capturing artificial viscosity $\hat{\epsilon} = h^2|u_{h,t} + u_h u_{h,x}|$, cf. [23]. For fast convergence of multigrid iterations it is important to smooth the artificial viscosity $\hat{\epsilon}$, which we do by averaging $\hat{\epsilon}$ with the neighboring values. In the program to improve conservation properties we integrate by parts the convection terms of (17).

It is suitable to use a multigrid method to solve (16) and (17). In an explicit pseudo time-marching method, the length of the local time step is by stability reasons determined from the smallest length scale h. For an adaptively refined mesh the "time step" needs to be very small, in particular for anisotropic meshes, in which it is possible with few degrees of freedom to construct very small and thin elements. Therefore multigrid methods can be expected to be even more useful for anisotropic meshes compared with isotropic meshes.

We use the full approximation scheme (FAS), which is a multigrid method for nonlinear problems, see [11], [13]. Let us for the sake of completeness describe the FAS here.

We want to solve the finite element equation $F_h(u_h, \phi_i^h) = 0$ for all standard basis functions $\phi_i^h \in V_h$, cf. (16) and (17). As a start solution on a refined grid we choose an interpolation of the approximate solution $\tilde{u}_{2h}$ on the coarse grid and set $u_h^0 = \tilde{u}_{2h}$.

We make ν (pre) smoothing steps on this equation with the damped Jacobi method,

$$u_h^{n+1}(x_i) = u_h^n(x_i) - cF_h(u_h^n, \phi_i^h)/D_i^n, \quad n = 0, 1, \ldots, \nu - 1, \quad \forall \text{ nodal points } x_i, \tag{18}$$

where $c = 0.75$ is a damping constant and D_i^n is the lumped diagonal part of the assembled equation for node i. After the smoothing, the algebraic equation has a residual

$$F_h(u_h^\nu, \cdot) \ . \tag{19}$$

To solve the equation exactly we would like to find a correction $d_h \in V_h$ such that

$$F_h(u_h^\nu + d_h, \cdot) = 0 \ . \tag{20}$$

Because it is expensive to solve on the fine grid h, we seek a correction $d_{2h} \in V_{2h} \subset V_h$ to equation (20), such that

$$F_{2h}(u_h^\nu + d_{2h}, v) = 0, \quad \forall\ v \in V_{2h}.$$

The correction d_{2h} can be represented as the deviation

$$d_{2h} = \bar{u}_{2h} - u_{2h}$$

from a given function $\bar{u}_{2h}$. We choose $\bar{u}_{2h} = \tilde{u}_{2h}$, an alternative is to choose $\bar{u}_{2h} = \pi_{2h}(u_h^\nu)$, where π_{2h} is the nodal interpolant on V_{2h}. The function u_{2h} is the solution of a coarse grid problem

$$F_{2h}(u_{2h}, v) = F_h(u_h^\nu, v) - F_{2h}(\bar{u}_{2h}, v), \quad \forall\ v \in V_{2h}. \tag{21}$$

Solve (21) and update u_h with the correction:

$$u_h = u_h^\nu + d_{2h} = u_h^\nu + \bar{u}_{2h} - u_{2h}. \tag{22}$$

After this correction, post smoothing is performed by applying (18) with $u_h^0 = u_h$ from (22), and the new residual is computed as in (19). If this residual is sufficiently small, the finite dimensional problem is solved, otherwise the steps above are repeated.

Note that (21) on the coarse grid is of the same type as our original problem (but with another right hand side). Therefore, the procedure described above can be recursively repeated by solving (21) on coarser levels where it is cheaper to solve. The resulting algorithm is called a V-cycle.

4.4 Refinement criteria

We describe in this section the implementation of steps i), ii) and iii) in the basic mesh enrichment algorithm in Section 4.1 for our model problems.

Let us first explain the algorithm for Burgers' equation. For this case the estimate of the error at an edge is based on the interpolation error in maximum norm for piecewise linear finite elements described by Lemma 7.

We start by computing for each element the following two-by-two matrices based on the approximate solution u_h

$$H_0(u_h) \approx \begin{bmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{bmatrix}, \quad H_1 = \frac{1}{[u_h]} \begin{bmatrix} u_{h,x}u_{h,x} & u_{h,x}u_{h,y} \\ u_{h,y}u_{h,x} & u_{h,y}u_{h,y} \end{bmatrix}. \tag{23}$$

The quantity $[u_h]$ is the difference between the maximum and minimum of the solution in the set of neighbor triangles with a node in common with the triangle. The computation of the second derivatives in the approximate Hessian $H_0(u_h)$ is described in Section 4.5. Add the two matrices

$$H = \alpha_0 H_0 + \alpha_1 H_1, \tag{24}$$

where $\alpha_0 = (1+\eta), \alpha_1 = \eta$, and $\eta = \log(\hat{\epsilon}/\epsilon)$. The artificial viscosity $\hat{\epsilon}$ in (17) is chosen as $\hat{\epsilon} = \sqrt{\epsilon^2 + \tilde{\epsilon}^2}$, where $\tilde{\epsilon} = h^2|u_{h,t} + u_h u_{h,x}|$.

Motivated by the error estimate Lemma 7, we compute for each triangle in the mesh

$$E_i = |\vec{a}_i^t H \vec{a}_i|/8, \quad \text{for edges} \ \ i = 1, 2, 3. \tag{25}$$

The edges with $E_i > \min(0.2, \max_j(E_j)/8)$ are marked for refinement, where j runs over all edges in the mesh. The marked edges are subdivided into two by adding a new node at the midpoint. The rank one matrix H_1 is added to ensure that refinement is indicated at inflection points in, for example, a shock and that stretching of elements is done in the right direction.

We now describe the refinement criterion for the elliptic model problem. We have found that the algorithm described for Burgers' equation, aimed at L_∞-control, needs to be modified for an elliptic problem such as Poisson's equation, $-\Delta u = f$. In our algorithm the anisotropic elements can have large angles which may cause the approximation of gradients to deteriorate, cf. Section 3. To ensure that gradients are well approximated it is natural to base the criterion on Cea's lemma:

$$||u - u_h||_E \leq \min_{v \in V_h} ||u - v||_E, \tag{26}$$

which is valid also if the triangulation violates a maximum angle condition. The subscript E denotes the energy norm

$$||w||_E^2 \equiv \int_\Omega |\nabla w|^2 dx,$$

and V_h is the finite element space where we seek the solution. We want to choose the triangulation such that the right hand side of (26) becomes small and consequently also the energy-norm error of the approximate solution.

For piecewise linear finite elements an approximation to the right hand side of (26) is obtained by letting $v = u_h$ and approximate the exact solution u in each triangle by a quadratic polynomial. Our strategy is: compute $H_0(u_h)$ for each triangle K and evaluate $||p - u_h||_E(K)$, where p is the unique quadratic polynomial with Hessian $H_0(u_h)$ and $p(x_i) = u_h(x_i)$ for all nodal points x_i. The size of $||p - u_h||_E(K)$ will guide the refinement of the triangle. Similarly this can be generalized to quadratic elements by replacing the quadratic polynomial with a polynomial of degree three. The algorithm is in fact easier to explain for the quadratic case; it works as follows.

We start by assuming that we have approximations of the third derivatives of the solution. As in the algorithm for Burgers' equation, we want to control the approximation error at midpoints of edges, but now we use the error in the gradient instead of the solution itself. Let $(\tilde{x}, \tilde{y})$ be the Cartesian components of an edge considered as a vector in $\boldsymbol{R}^2$. The interpolation error of the gradient at the midpoint of the edge is approximated by

$$\Lambda = \sqrt{\lambda_1^2 + \lambda_2^2},$$

where

$$\lambda_1 = (u_{xxx}\tilde{x}^2 + 2u_{xxy}\tilde{x}\tilde{y} + u_{xyy}\tilde{y}^2)/8,$$
$$\lambda_2 = (u_{xxy}\tilde{x}^2 + 2u_{xyy}\tilde{x}\tilde{y} + u_{yyy}\tilde{y}^2)/8.$$

The quantity Λ determines the edges to be refined similarly to as E_i in (25) was used for Burgers' equation.

The refinement algorithm is as follows: First compute $\max(\Lambda)$, where the maximum is taken over all edges of the mesh. Then, refine the edges for which $\Lambda > \max(\Lambda)/c$, with

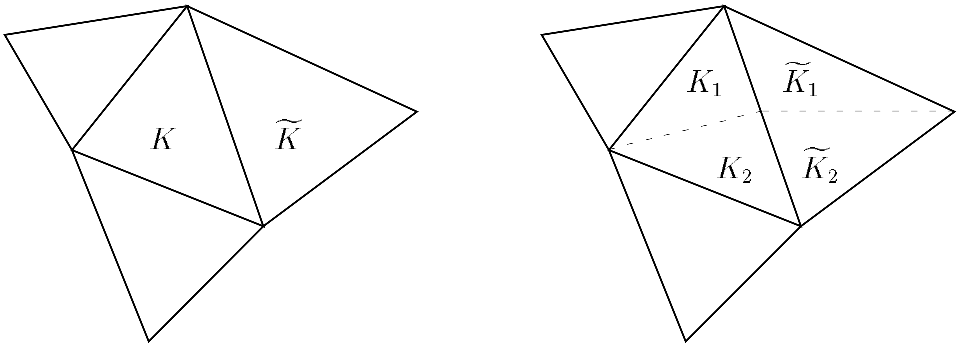

Figure 4: Triangles for refinement in linear elliptic problem.

factors $c = 4$ and $c = 6$ for isotropic and anisotropic refinement, respectively. Elements with large quadrature errors are also refined, see Remark in Section 5.2. For the possible new interior edges the alternative with least value of Λ is chosen.

Let us now study the refinement algorithm for the linear elliptic test problem, where the algorithm is more complicated and has some ad hoc constructions. The reason for the complications is that refinement of a triangle affects the neighbor triangles so that the refinement decision can not be made as local as was possible for the quadratic elements, cf. Fig. 1. Assume that an approximation of $H_0(u_h)$ the Hessian of the solution has been computed. The error indicator approximating the local energy-norm error is

$$\tilde{B}^2(K) \equiv \int_K |\nabla p|^2 dx = \frac{|K|}{3} \sum_{i=1}^{3} |\nabla p(y_i)|^2, \tag{27}$$

where the $|K|$ denotes the area of triangle K and y_i the edge midpoints. The function p is the unique quadratic polynomial which is zero at the vertices of K and has Hessian $H_0(u_h)$. The equality in (27) follows from the exactness of the edge midpoint quadrature rule for quadratic functions. We use $\tilde{B}(K)$ as a local approximation in the right hand side of (26), and it is computed for both the Hessian $H_0(u_h)$ and the rank one matrix, $H_1 = 0.06(u_{h,x}^2, u_{h,x}u_{h,y}; u_{h,y}u_{h,x}, u_{h,y}^2)$. The quantity error $\tilde{B}(K)$ is computed separately for H_0 and H_1 and then added; the sum is denoted by $B(K)$. Let $\widetilde{K}$ be a neighbor triangle to the triangle K, and K_1, K_2 the subtriangles of K obtained by bisection of the edge shared by K and $\widetilde{K}$, see Fig. 4. To decide if it would be advantageous to divide any of Ks edges we compute the ratio

$$E = \frac{B(K) + B(\widetilde{K})}{B(K_1) + B(K_2) + B(\widetilde{K}_1) + B(\widetilde{K}_2)} .$$

The quantity E is computed for the three edges of K. A large value of E indicates that the gradient error is likely to decrease if the edge is divided.

Let the breakpoint for refinement be $TOL = \min(10, \max_K(B(K))/6)$. Refine all edges which have $B(K) > TOL$ and $E > 1.1$. If $B(K) > TOL$, and no edge of K is marked, then the two edges with largest ratio E are marked anyway to force refinement. This heuristic is chosen since for two marked edges there are more alternatives to find an acceptable refinement compared to the case that only one edge had been marked.

As usual, refined triangles are given $A1, A2$ or $A3$ refinement status. When two or three edges are to be refined, it is important to choose the best of the internal triangle subdivision alternatives, based on a criterion similar to E above.

Remark: The main reason why the algorithm is complicated in the linear elliptic case, is the difficulty to guarantee that the error decreases when one triangle is bisected. It is necessary to take into account that the bisection affect the neighbor triangles.

4.5 Computation of higher derivatives

In the refinement criteria we need to approximate the derivatives of the solution to one degree higher than is given directly by the degree of the polynomial of the used finite element. In this section we describe how these approximations are done. For piecewise linear finite elements, a nodal approximation of the Hessian can be computed by partial integration over the support of the basis function, cf. [7], [24, 25]. The L_2-projection of second derivatives $D_{xx}u_h \in V_h$ is defined by

$$\int_\Omega D_{xx}u_h v dx = -\int_\Omega u_{h,x}v_x dx, \quad \forall v \in V_h. \tag{28}$$

Lumping of the mass matrix from (28) gives an approximation to $D_{xx}u_h$ at node i by

$$\tilde{D}_{xx}u_h^i \equiv -\frac{1}{3|S_i|}\int_{S_i} u_{h,x}\phi_x^i dx, \tag{29}$$

where the domain S_i is the union of the triangles that have node i in common, and ϕ^i is the standard nodal basis function related to node i. The derivatives u_{xy} and u_{yy} are computed in a similar way.

The approximation of gradients using piecewise linear elements can deteriorate for elements with large angles, cf. Section 3. This is confirmed in experiments where (29) is used to compute the Hessian of a known function on an anisotropic grid; the accuracy of the computed Hessian is low. We have found that the accuracy can be improved by using "virtual edge swapping" in the computation of (29), that is, edges where the sum of the opposing angles is larger than π are diagonally swapped, see Fig. 5, and then (29) is applied with the associated basis function. Observe that this is done temporary and only to compute the Hessian. We have also experienced that the accuracy is improved by smoothing the computed Hessian, e.g., by averaging over neighbor nodes.

Next we describe the L_2-projection for third degree derivatives. For $D_{xxx}u_h \in V_h$ where V_h is the space of piecewise linear functions, the L_2-projection is defined by:

$$\text{Find } D_{xxx}u_h \in V_h \text{ such that} \quad \int_\Omega D_{xxx}u_h \; v dx = -\int_\Omega u_{h,xx}v_x dx, \quad \forall v \in V_h. \tag{30}$$

For quadratic elements it is not possible to use virtual edge swapping as described above, since the nodal values at midpoints of edges can not be used after swapping. Instead we

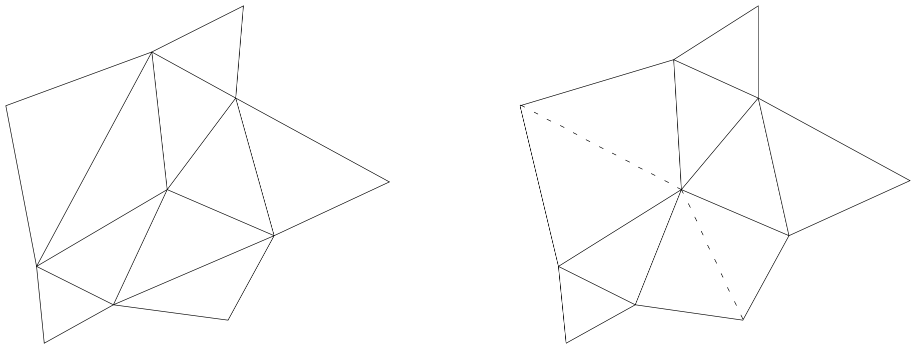

Figure 5: Before and after virtual edge swapping.

compute the L_2-projection without doing any lumping. To smooth the solution and avoid Gibb's phenomenon and we add a small artificial diffusion term of the form

$$0.7 \int_\Omega h|K|^{3/2} \nabla u_{h,xxx} \nabla v dx$$

to the left hand side of (30). The resulting system of equations has a bounded condition number and therefore we can solve it accurately enough with a few damped Jacobi iterations, described by (18).

5 Numerical experiments

5.1 Burgers' equation

To ease the reproduction and validation of results it is important to use model problems where exact solutions is known. We use Burgers' equation

$$u_t + (\frac{u^2}{2})_x = \epsilon u_{xx}, \tag{31}$$

with $u(x,0) = u_0(x)$ and $\epsilon > 0$, which can be explicitly solved by the Hopf-Cole transformation [14]. Thus we get

$$u(x,t) = \frac{\int_{-\infty}^{\infty} \frac{x-y}{t} e^{-v_0(y)/2\epsilon} e^{-(x-y)^2/4\epsilon t} dy}{\int_{-\infty}^{\infty} e^{-v_0(y)/2\epsilon} e^{-(x-y)^2/4\epsilon t} dy}, \tag{32}$$

where

$$v_0(y) = \int_{-\infty}^{y} u_0(z) dz. \tag{33}$$

When the initial data $u_0(x)$ is piecewise linear, the integral can be solved in terms of the error function erf. We avoid a discontinuous solution by setting the computational domain to $(x,t) \in (0,1) \times (0.01, 1.01)$. The initial data for our Burgers problem is given in Table 1. The initial conditions are chosen such that the characteristics at the boundaries $x = 0$ and $x = 1$ are directed into the domain. The initial and boundary conditions are supplied as

interval	$u_0(x)$
$(-\infty,-5)$	0
$(-5,0.2)$	$2+1.4x$
$(0.2,0.8)$	0.8
$(0.8,3)$	$-2+0.1x$
$(3,\infty)$	0

Table 1: Initial data for the Burgers model problem at $u_0(x)=u(x,0)$.

Dirichlet conditions using (32). At the time $t=1.01$ a homogeneous Neumann condition is applied. We have chosen data for $u_0(x)$ such that the solution has interacting curved shocks, see Fig. 6, other choices can give rarefaction waves.

Our problem can either be seen as a one-dimensional space-time problem or a stationary two-dimensional problem. Some advantages with space-time discretization of Burgers' problem is the possibility to use elements with high aspects ratios in the time direction and that multigrid can be used also for transient problems. The obvious disadvantage is the increased complexity to generate a mesh of one dimension higher than the ordinary space discretization and the associated need for more degrees of freedom.

5.2 Elliptic model problem

Our elliptic model problem is the Poisson equation

$$-\Delta u = f, \quad \text{in} \quad \Omega,$$

$$u = g, \quad \text{on} \quad \partial\Omega,$$

with the right hand side f and boundary data g chosen such that the solution is

$$u=\tanh(\frac{r-a}{\epsilon}), \quad \text{where} \quad r=|x-x_0|, \quad a=0.75, \quad x_0=(0.2,0.3), \quad \epsilon>0, \tag{34}$$

and Ω is the unit square. The solution (34) has an interior layer of width ϵ, see Fig. 7.

5.3 Computations

We will now describe results of numerical experiments for the two model problems. Table 2 shows the set of problems which has been solved. The isotropic algorithm is used for the first refinements of an initial mesh with two triangles until the main structure of the solution is found. Fig. 6 shows solution contour plots and the adaptive mesh for the Burgers model problem with data from Table 1. Note that the refinement for the (well resolved) solution is coarser at the inflection point of the shock, as is expected from an approximation point of view for linear elements.

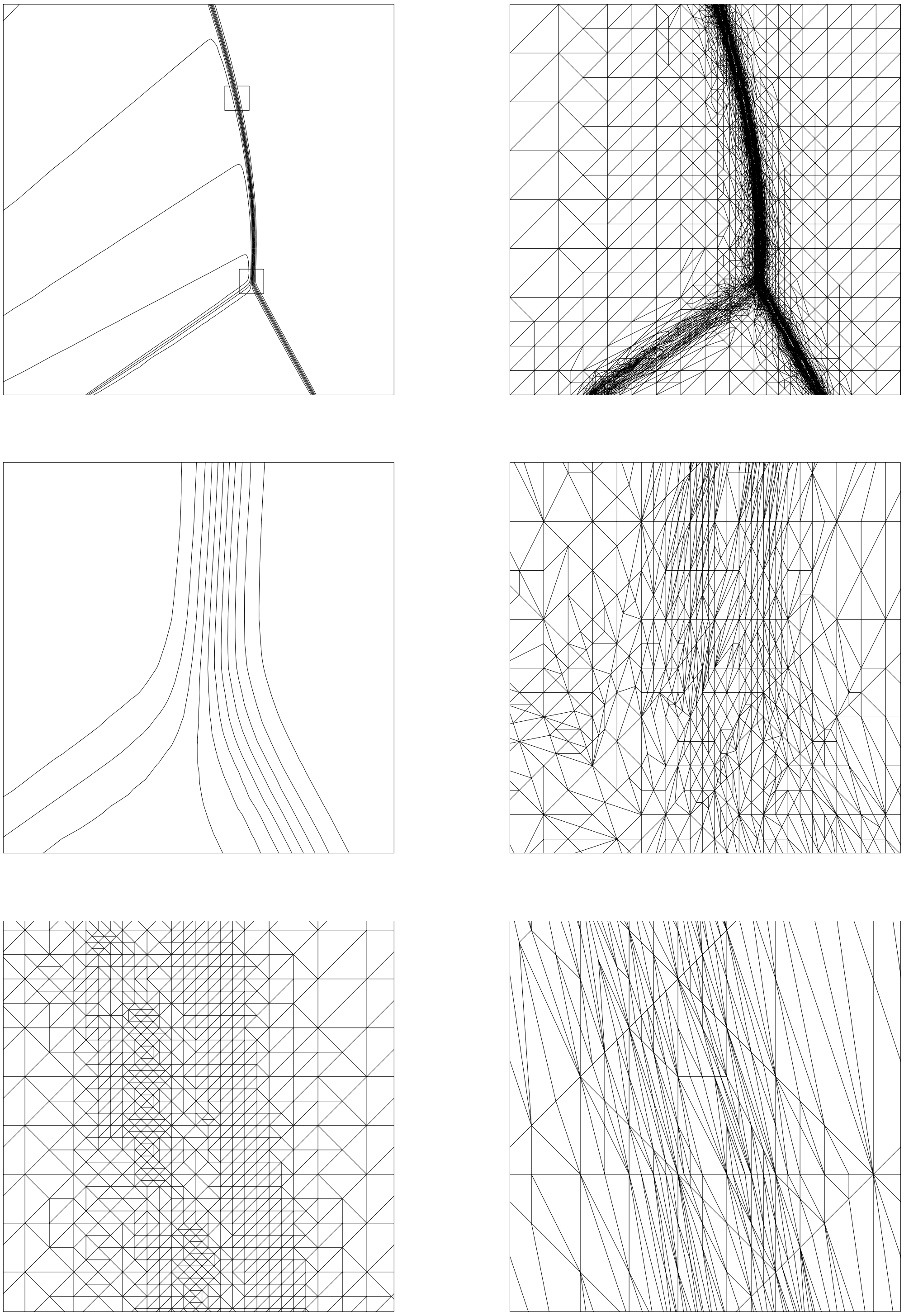

Figure 6: The Burgers model problem with $\epsilon = 0.005$ using linear elements. Top: solution and the corresponding anisotropic mesh after 12 refinements using 10190 triangles. In the middle: close up of the solution and mesh at the shock intersection. Bottom: close up at the upper box in the solution, isotropic after 10 refinements using 23901 triangles, and anisotropic after 12 refinements. The magnification factor is 16.

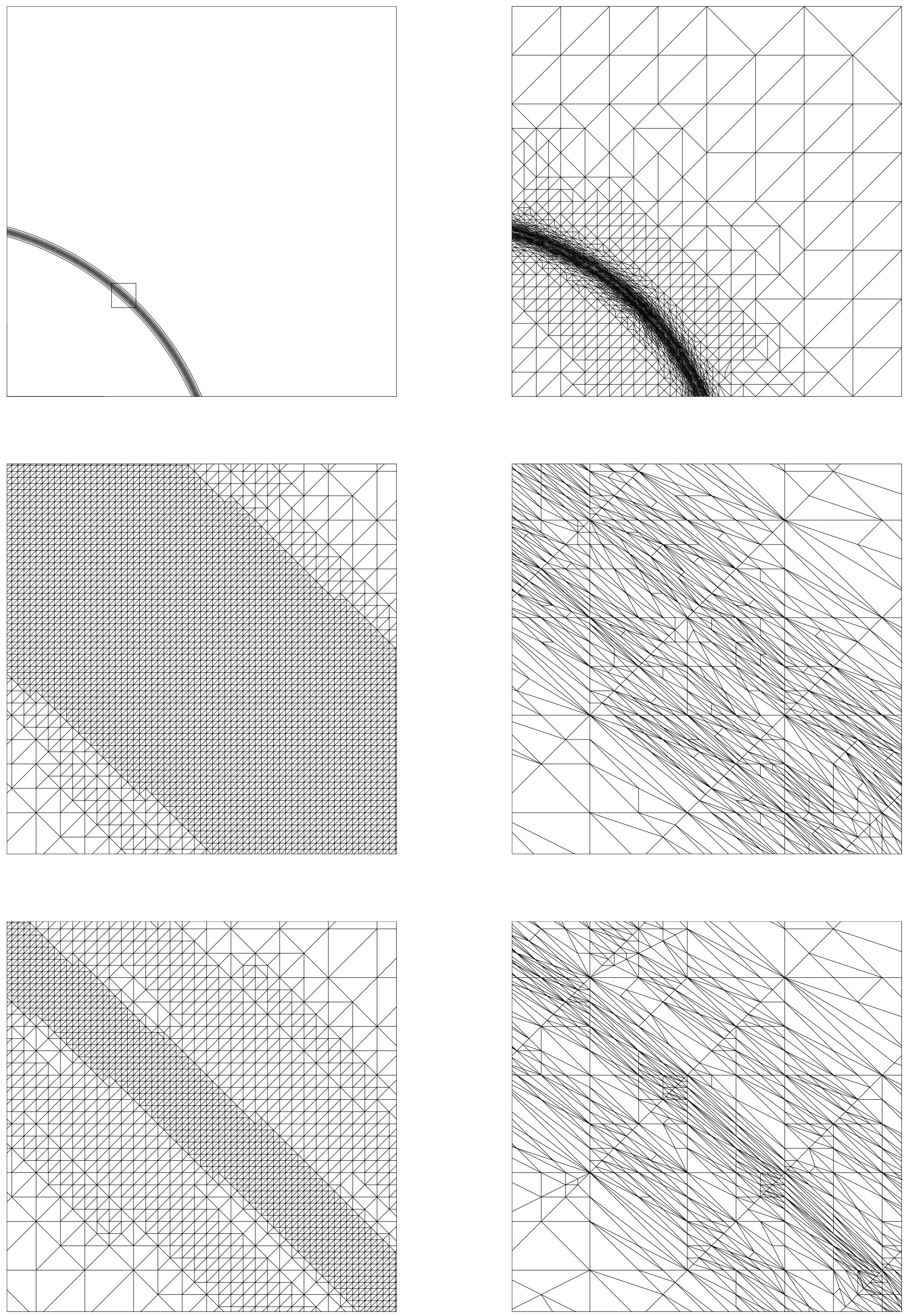

Figure 7: Solution and meshes of the Poisson model problem with $\epsilon = 0.01$. Top: solution and the corresponding anisotropic mesh with 5617 triangles after 10 refinements. The four remaining figures shows 16 times magnification at the box in the solution. In the middle using linear elements: isotropic with 70024 triangles after 10 refinements and anisotropic with 10757 triangles after 12 refinements. Bottom using quadratic elements: isotropic after with 30205 triangles after 10 refinements and anisotropic after with 9304 triangles after 12 refinements.

Burgers linear	Poisson linear	Poisson quadratic
0.002	0.005	0.002
0.005	0.01	0.005
0.01	0.05	0.01
0.02	0.1	0.02

Table 2: The set of used ϵ in numerical experiments. The parameter ϵ is viscosity and width for the Burgers and Poisson equations, respectively. All problems have been solved with both isotropic and anisotropic algorithms.

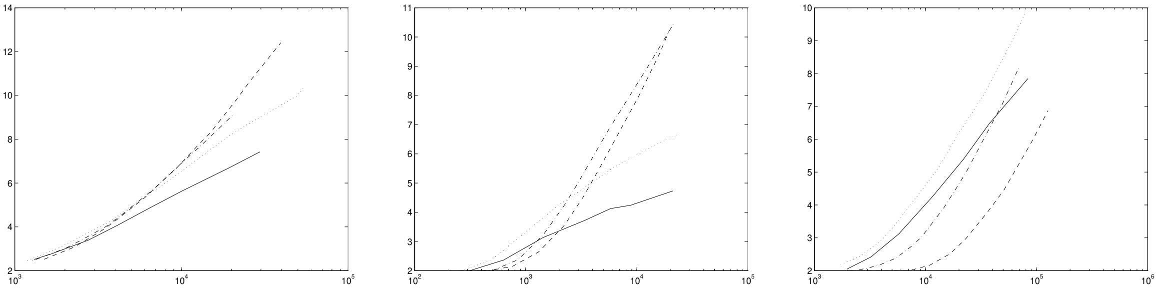

Figure 8: Mean aspect ratio of triangles as a function of degrees of freedom for the sequence of anisotropically refined meshes. From the left Burgers with linear elements, Poisson with linear elements and Poisson with quadratic elements. Solid line graphs for the largest value of ϵ in Table 2, then dotted, dash-dot and dash-dash graphs for decreasing values of ϵ.

In Fig. 7 we see graphics for Poisson's equation with the solution (34) for piecewise linear and quadratic elements. For quadratic elements we see that there are two positions in the "shock" where small third derivatives of the solution makes the approximation easier.

We define the aspect ratio of a triangle as the ratio between the large and small length-scale, H/h, which yields the aspect ratio 2 for a $\pi/2, \pi/4, \pi/4$ triangle. However, although the elements have the aspect ratio 2 they do not in general use their direction; in numerical experiments the "practical" aspect ratio seems to be $\sqrt{2}$. The evolution of the mean aspect ratio as the for the anisotropically refined meshes is seen in Fig. 8. The increase of the mean aspect ratio is larger for smaller values of ϵ, which also results in larger values for the final mesh with the exception for quadratic elements. As can be expected, the solutions with small ϵ can better use the larger geometric flexibility given by anisotropic refinement. Fig. 9 shows for the final meshes a logarithmic distribution of the aspect ratio of triangles.

We see from the pictures of triangulations in Figures 6, 7 and from Figures 8, 9 that it is possible to generate highly anisotropic triangulations with our algorithms. Crucial questions are now: What are the errors for the anisotropic meshes compared to isotropic meshes? Do we gain anything by using anisotropic refinement? In Figures 10 and 11, we graph the errors $||u - u_h||$ for the Burgers and Poisson model problems, using both isotropic and anisotropic

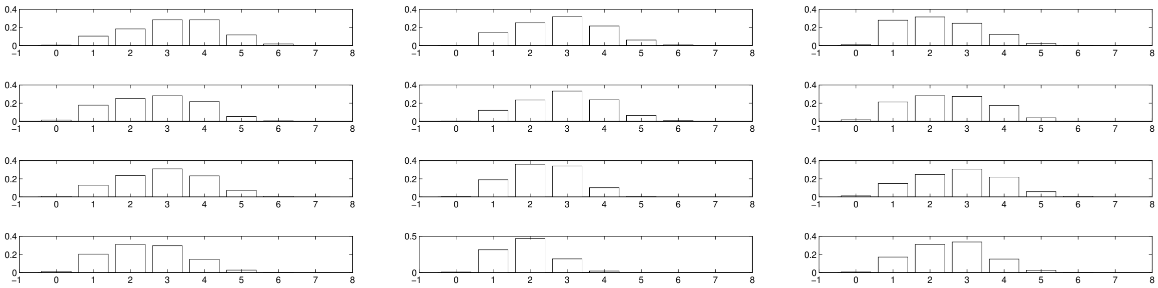

Figure 9: Distribution of aspect ratio of triangles in logarithmic scale, more precisely the integer part of $\log(H/h)/\log(2)+0.5$. From the left: Burgers linear elements, Poisson linear elements and Poisson quadratic elements. Bottom to top show increasing values of ϵ from Table 2.

refinement algorithm.

Remark: It is important to include the quadrature error of the right hand side in the finite element problem. For our elliptic model problem it is, due the quadrature error, not useful with anisotropic refinement as long as the quadrature error is larger than the Galerkin approximation error, which is approximated by $\tilde{B}$ in (27). This is in contrast to Burgers' problem of hyperbolic type, where artificial viscosity gives a smooth solution even for a coarse mesh; the anisotropic refinement can be used before the viscous profile is resolved. It is sufficient that the main structure of the solution is correct and hence it is possible to extract directions of refinement.

We estimate the size of the local quadrature error by using two different sets of quadrature rules and use the difference of the evaluations as the estimate of the error. The element is isotropically refined as long as the quadrature error is larger than the approximation error.

6 A model approximation problem

In this section we compare the approximation error of the finite element solution to the interpolation error of the exact Burgers solution with the isotropic and anisotropic mesh refinement algorithms.

We use the Burgers problem studied in the numerical experiments, with the exact solution (32) and data given in Table 1. This Burgers solution is a suitable test case; it contains anisotropy in the form of interacting curved shocks. Using the exact solution, the maximum interpolation error at an edge e is estimated by computing $M_e = \max_i |(u-\pi u)(x_i)|$, where x_i are points at positions $0.25, 0.5, 0.75$ along the edge. The anisotropic and isotropic refinement algorithms, in Section 4.1, are applied replacing the quantity E_e in (15) with M_e, and refining all edges with $M_e \geq 0.25 \max_{\tilde{e}} M_{\tilde{e}}$. The error in L_∞-norm is evaluated at six quadrature points in each triangle.

In Fig. 12 we compare the pure approximation algorithm with the finite element method and conclude that the gain in degrees of freedom using the two anisotropic refinement methods are similar.

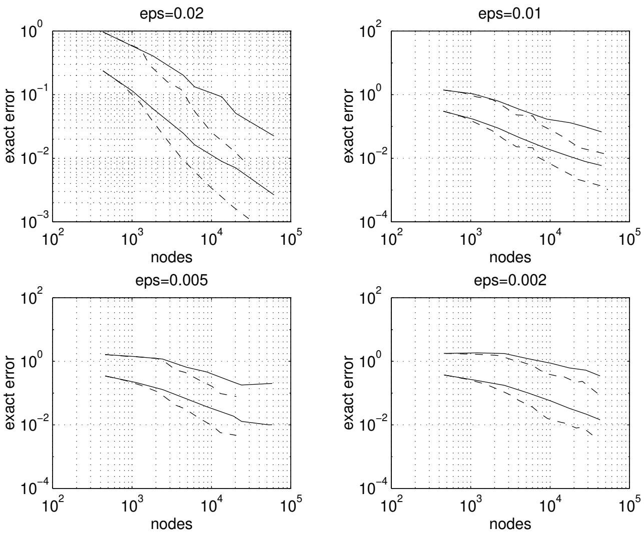

Figure 10: Convergence of solution for Burgers equation. Solid and dashed lines show the error of the solution for isotropic and anisotropic refinement, respectively. The upper pair of graphs are the error in L_∞-norm and the lower in L_2-norm.

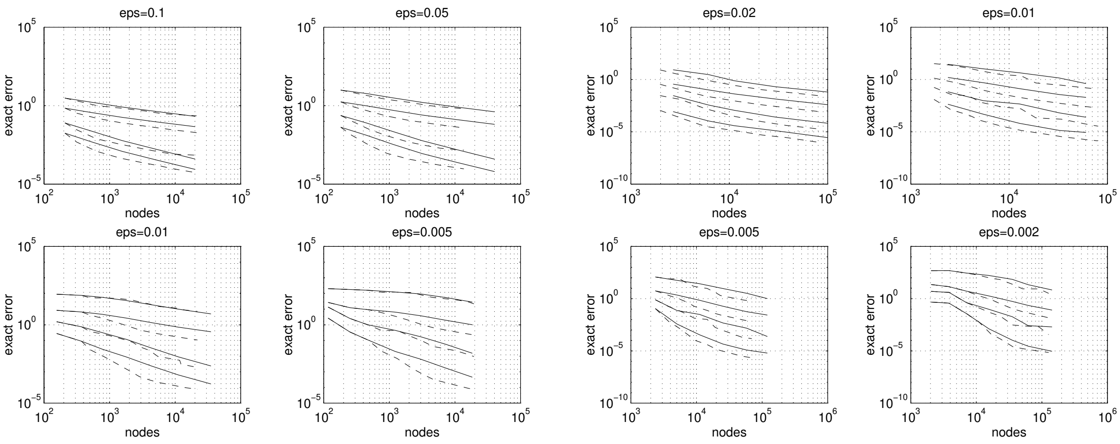

Figure 11: Error convergence for the Poisson equation. To the left linear elements, and to the right quadratic elements. Errors for isotropic and anisotropic refinement are shown with solid and dashed lines, respectively. From top to bottom in each diagram the errors in $W^1_\infty, W^1_2, L_\infty$ and L_2-norms are shown, where W^n_p denotes the Sobolev space of functions with n derivatives in L_p.

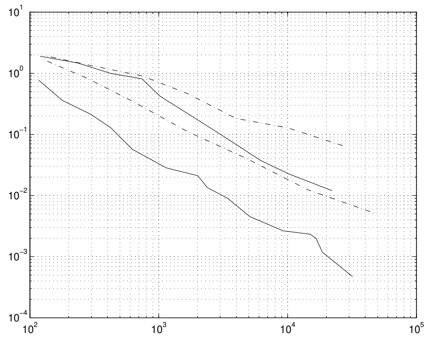

Figure 12: Error convergence for the Burgers equation with $\epsilon = 0.01$ using linear elements. The graphs show the error in L_∞-norm versus the degrees of freedom for the pure approximation algorithm, upper two graphs, and the finite element method, lower two graphs. Isotropic refinement are shown with dash-dotted lines and anisotropic with solid lines. The refinement criterion used for the finite element method is that the midpoint error indicated by separately by the two matrices $(1+\eta)H_0$ and $0.4(0.5+\eta)H_1$ are added, see (24) and (25).

Remark: In the middle of the curved shock the Hessian of the solution has rank 1 and in fact indicates stretching in the "wrong" direction. In the pure approximation problem this is fixed by sampling the interpolation error at three points on each edge. The anisotropic algorithm does not work if only the midpoint is used. For the finite element algorithm our remedy to this problem is to include error indicated from the rank 1 matrix H_1, cf. (23). The normalization for H_1 is motivated by the interpolation error at an edge for $u(x,y) = x^3$.

7 Conclusions

It is difficult to find a good refinement algorithm for piecewise linear elements. The reason is that it is hard to avoid large angles for which the gradient error becomes large. To allow more freedom in the placement of new nodes, e.g., by not restricting new nodes to be at midpoints of edges, could lead to better performance, but at the expense of more complicated algorithms.

Using quadratic elements looks more promising than linear elements, at least in numerical experiments it seems possible to get a reasonable convergence rate even if the angles are large; the algorithm is more robust. Practically it is very important that the error decreases when an element is bisected, then it is not necessary to take the effect of neighbors into consideration when a triangle is refined. This local aspect makes the algorithm easier and cheaper to implement.

For convection dominated problems the effect of large angles has less influence compared to the linear Poisson problem. In a sense only one half derivative of the solution is necessary for good approximation, in the convection problem (17), for small ϵ whereas the problem

(16) needs good approximation of one derivative, cf. Cea's lemma (26).

We have implemented a multigrid method which works also for the unconventional meshes generated by our anisotropic algorithm. A priori it is not at all obvious that the multigrid method should work for an anisotropic mesh. We have found that the required number of V-cycles is larger by approximately a factor of two for anisotropic meshes by a factor of two approximately than the corresponding isotropic cases. We have seen that our anisotropic refinement algorithm can generate meshes with fewer degrees of freedom than the conventional isotropic algorithm. The factor of saved degrees of freedom is between five and ten, for meshes yielding an error of order 10^{-2}.

References

[1] T. Apel and M. Dobrowolski, *Anisotropic interpolation with applications to the finite element method*. Computing, **47**, (1992), pp. 277–293,

[2] T. Apel and G. Lube, *Local inequalities for anisotropic finite elements and their application to convection-diffusion problems*, Preprint SPC94_26, TU Chemnitz-Zwickau (1995)

[3] T. Apel and S. Nicaise, *Elliptic problems in domains with edges: anisotropic regularity and anisotropic finite element meshes*, Preprint, SPC94??, TU Chemnitz-Zwickau (1994)

[4] E. F. D'Azevedo and R. B. Simpson, *On optimal triangular meshes for minimizing the gradient error*, Numer. Math. **59**, (1991), pp. 323-348

[5] E. F. D'Azevedo and R. B. Simpson, *On optimal interpolation triangle incidences*, Siam J. Sci. Stat Comput. Vol. 10, No. 6, (1989), pp. 1063-1075,

[6] I. Babuŝka and A. K. Aziz, *On the angle condition in the finite element method*, SIAM J. Numer. Anal., **13**, (1976), pp. 214–226

[7] T. J. Barth *Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations*, AGARD-R-787 Special course on unstructured grid methods for advection dominated flows, (1992)

[8] R. E. Bank *PLTMG, a software package for solving elliptic partial differential equations: users' guide 7.0.* SIAM, Philadelphia, (1994).

[9] R. Becker, C. Johnson and R. Rannacher, *An error control for multigrid finite element methods*, Preprint 1994-36, Department of Mathematics, Chalmers University of Technology, Göteborg

[10] J. H. Bramble, *Multigrid Methods*, Pitman research notes in mathematical sciences, (1993)

[11] A. Brandt, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, GMD-Studien Nr. 85, (1984)

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

[13] W. Hackbush, *Multigrid Methods and Applications*, Springer-Verlag, (1985).

[14] E. Hopf, *The partial differential equation $u_t + uu_x = \mu_{xx}$*, Comm. Pure Appl. Math. **3**, pp. 201–230, (1950)

[15] Claes Johnson, *Numerical solution of partial differential equations by the finite element method.* Studentlitteratur, Lund, (1987)

[16] C. Johnson, U. Nävert and J. Pitkäranta, *Finite element methods for linear hyperbolic problems,* Comp. Meth. Appl. Mech. Eng., **45** (1984), pp. 329-336

[17] C. Johnson and A. Szepessy, *Adaptive finite element methods for conservation laws based on a posteriori error estimates*, Comm. Pure Appl. Math., 98 (1995), pp. 199–234

[18] M.C. Rivara, *Design and data structure of fully adaptive, multigrid, finite element software*, ACM Trans. Math. Software **10** (1984), pp. 242–264

[19] M.C. Rivara, *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques*, Int. J. Numer. Meth. Engrg. **20** (1984), pp. 745–756

[20] K. Samuelsson, *Adaptive Algorithms for Finite Element Methods Approximating Flow Problems*, Thesis, NADA KTH, Stockholm to appear.

[21] K. Samuelsson and A. Szepessy, *On the convergence of multigrid methods for flow problems*, Preprint TRITA-NA-9511, NADA KTH, Stockholm.

[22] K. G. Siebert , *An a posteriori error estimator for anisotropic refinement*, SFB 256 Bonn, Preprint no. 313 (1993), to appear in Numer. Math.

[23] A. Szepessy, *Convergence of a shock-capturing streamline diffusion finite element method for a scalar conservation law in two space dimensions,* Math. Comp., **53** (1989), pp. 527–545

[24] J. Z. Zhu, O. C. Zienkiewicz, *Adaptive techniques in the finite element method,* Commun. Appl. Numer. Methods **4** (1988), pp. 197–204

[25] O. C. Zienkiewicz, J. Z. Zhu, *A simple error estimator and adaptive procedure for practical engineering analysis,* Int. J. Numer. Meth. Engrg. **24** (1987), pp. 337–357