A Comparison of Some Error Estimators for Convection-Diffusion Problems on a Parallel Computer

Volker John
Otto von Guericke Universität Magdeburg
Institut für Analysis und Numerik
Postfach 4120
D-39016 Magdeburg
Germany

Abstract

Adaptive techniques has been proved to be a necessary tool for the numerical treatment of partial differential equations. In order to reduce the size of the arising linear systems more computational work is put only into subregions where a large local error of the current discrete solution is assumed. A-posteriori error estimators are used to choose these subregions.

We consider some convection-diffusion problems in 2d and compare the results of the adaptive algorithms using different error estimators. We only choose the mesh size adaptively. The criteria of comparison are the reduction of the error in some norm, the time used for doing the estimate, and the parallel behaviour. We only choose such error estimators which work local and therefore are well fitted for the implementation on a parallel computer. All error estimators depend on a set of parameters. The comparison take place with parameters which seems to be reasonable *before* the solution process, i.e. we do not compare with 'optimal' sets which are chosen after having some experience with the test examples.

We present the results of some recent numerical tests on a parallel computer with a MIMD architecture and 128 processors. No one of the chosen error estimators was in general superior or interior than the others, even those which have theoretical shortcomings. All estimators worked well with general sets of parameters but there is to be expect a large improvement using other adaptive methods (grid alignment), too. We found that in general it is not enough only to refine the regions where a large error is assumed. A certain over-refinement in a neighbourhood of those regions often yields better results with respect to accuracy of the solution. Because there is some communication in the estimating process the error estimators have a modest lost of parallel efficiency.

1 Introduction

Let be given a partial differential equation in a region Ω and let u_0 be the solution. In addition let $u_{h,p}$ be the finite element solution of the discretized partial differential equation corresponding to a (regular) mesh with mesh function $h(x)$ and piecewise polynomial approximation of order p.

We want to find a discretization and a discrete solution $u_{h,p}$ such that for a given tolerance TOL and a given norm $\|\cdot\|$ the error bound

$$\|u_0 - u_{h,p}\| \leq TOL \tag{1}$$

is satisfied and we want to spend as less computational work as possible to achieve this. The computational work can be measures in the number of unknowns or in CPU-time. To find a mesh with a minimal number of unknowns is a complex optimization problem and it will be solved in practice approximately. The tools for doing that are adaptive methods and an important part of this methods are error estimators.

Error estimators have two tasks:

1. They should indicate such subdomains of Ω where a large error occurs in order to put more computational work into these regions. Let η_{loc} be the local error estimate then an inequality like

$$\eta_{loc} \leq c\|u_0 - u_{h,p}\|_{loc}, \quad c > 0, \tag{2}$$

   insures that there exists a local error if the error estimator indicates a local error.

2. Error estimators can be used as a termination criterion for the computational process if an estimate like

$$\|u_0 - u_{h,p}\| \leq C\eta_{glob}, \quad C > 0$$

   holds, where η_{glob} is the estimate of the global error and C is known approximately and it is not too large.

There are several adaptive methods, e.g.

- the adaptive choice of the grid size, the h-method
- the adaptive choice of the degree of the local approximation, the p-method
- grid alignment, grid orientation, grid stretching
- combined methods

Besides error estimators there exist so-called error indicators. Both, error estimators and indicators, have the ability to indicate subregions where a large error occurs. But only the estimators say how large the error is and can be used for the second task. Generally error indicators are cheap and easy to implement.

In this paper as a first step we only consider the behaviour of some error estimators with respect to the first task and to the adaptive h-method. That's why we include some error indicators in the comparison, too. A comparison with respect to the second task and for the Stokes equations can be found in [BW90]. The first criterion for our comparison is the accuracy of the obtained discrete solution on the mesh produced by the different error estimators. The second criterion is the performance in a parallel code (time, memory requirements).

There are a lot of open questions and problems concerning error estimators. We only want to state two.

1. One wants to control and estimate the error in some norm, see (1). The norm depends on the regularity of the solution u_0 which is in general not known a-priori. How does the result depends on the chosen norm ?

2. It is known that estimating even with the exact error in some norm in general does not produce an optimal mesh in the sense of computational work and accuracy. Sometimes the result is far beyond from being optimal, see e.g. [Mit89]. Is there a transport of the error and what can we do ?

This paper does not give answers on this questions. But here are presented examples which emphasize the importance to search ones.

The aim of our numerical work is the implementation of adaptive techniques into a parallel software package for the solution of several kinds of partial differential equations. A main feature is that these adaptive methods should be used by people who are not familiar with these techniques. The error estimators depend on a set of parameters. This set can be chosen a-priori and we demand that the error estimators should produce good results on a wide range of parameters. Therefore we do not turn our main consideration on finding the best possible set of parameters, which depends on the estimator and on the problem. Instead we investigate the behaviour of the estimators with some parameters which seem reasonable before the solution process and which probably are not fitted optimal to any of the error estimators or test problems.

The paper is organized as follows. In the second chapter we introduce some error indicators and estimators. In the third chapter we define some test problems and present the results of numerical tests. This is the main part of the paper. The fourth chapter is depicted to some concluding remarks.

2 Some Error Indicators and Error Estimators

2.1 The Model Problem

We consider the model problem

$$\begin{array}{rcll} -\nabla\cdot\varepsilon\nabla u + b\cdot\nabla u &=& f & \text{in} \quad \Omega \\ u &=& g & \text{on} \quad \Gamma_D \\ \varepsilon\frac{\partial u}{\partial n} &=& z & \text{on} \quad \Gamma_N \end{array} \tag{3}$$

where $\varepsilon > 0$ and $\text{meas}(\Gamma_D) > 0$. The discrete operators and functions we denote with a subscript h. The boundary $\partial\Omega$ is sufficiently smooth (Lipschitz continuous) and $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$ holds.

2.2 Two Simple Error Indicators

A very simple idea of a mesh-refinement strategy is to refine in such regions where the calculated approximation u_h of the solution u_0 is not 'sufficiently constant'. If there exists any interesting subregion then it is outside of the nearly constant subregions.

In one dimension let u_h be given in an interval $[a, b]$. The error indicator tests only if

$$|u_h(a) - u_h(b)| \geq TOL$$

in order to decide if the element will be refined or not. In more dimensions we use the same formulae. Let $\{x_i\}$ be the set of the nodes of an Lagrange finite element T. Then we calculate as an error indication

$$\eta_{M,T} = \max_{i \neq j} |u_h(x_i) - u_h(x_j)|.$$

Of course we can use instead of the maximum norm each other discrete norm.

An often used error indication is to compute the gradient of the current approximation of the solution in each element and scale it by the size of the element. We compute for a linear function in an element T the gradient in the barycentre S of this element and scale it by the diameter h of the element (longest edge), i.e. the error indication is

$$h\|\nabla u_h(S)\|_{l^2},$$

where $\|\cdot\|_{l^2}$ is the usual Euclidian vector norm. For general functions and d-dimensional elements a generalization of the error indication is

$$\eta_{G,T} = h^{(2-d)/2}\|\nabla u_h\|_{L^2,T}.$$

Lemma 2.1 *Let be given a regular triangulation of affine equivalent elements and a finite dimensional space of functions defined on each element . Then the error indicators $\eta_{M,T}$ and $\eta_{G,T}$ are equivalent, i.e. there exist constants c_1 and c_2 independent of the element T with*

$$c_1\eta_{G,T} \leq \eta_{M,T} \leq c_2\eta_{G,T} \quad \forall T.$$

Proof . Let $\hat{T}$ be the reference element and $\hat{P}$ the space of functions over $\hat{T}$. The mappings

$$\hat{u}_h \to \|\hat{\nabla}\hat{u}_h\|_{L^2,\hat{T}}$$

and

$$\hat{u}_h \to \max_{i,j} |\hat{u}_h(\hat{x}_i) - \hat{u}_h(\hat{x}_j)|$$

are norms on the quotient space $\hat{P} \setminus \mathbf{R}$. Because the equivalence of norms in finite dimensional spaces there exist constants c and C such that

$$c\|\hat{\nabla}\hat{u}_h\|_{L^2,\hat{T}} \leq \max_{i,j} |\hat{u}_h(\hat{x}_i) - \hat{u}_h(\hat{x}_j)| \leq C\|\hat{\nabla}\hat{u}_h\|_{L^2,\hat{T}}, \quad \forall \hat{u}_h \in \hat{P}$$

holds. Let T be an arbitrary element. Then there exists an affine transformation $B_T : \hat{T} \mapsto T$ with

$$\|B_T\| \leq c_1 h_T, \quad \|B_T^{-1}\| \leq c_2 h_T^{-1}, \quad c_3 h_T^d \leq |\det(B_T)| \leq c_4 h_T^d,$$

(see e.g. [Cia78]). Thus, by usual scaling arguments we obtain

$$c\|\nabla u_h\|_{L^2,T} \leq \max_{i,j} |u_h(x_i) - u_h(x_j)| \leq C\|\nabla u_h\|_{L^2,T}, \quad \forall u_h \in P.$$

■

Remark 2.2

- It is possible that the error indicator $\eta_{M,T}$ indicates an error where no error is, i.e. it does not fulfill (2). Let the solution u_0 of a problem be linear but not constant and let $u_h = u_0$. Then the error is zero for all elements but $\eta_{M,T}$ is not.

  It is also possible that $\eta_{M,T}$ indicates no error at all although the solution is not obtained. A (rather pathological) example is that u_h is constant but u_0 is not.

  The same holds of course for $\eta_{G,T}$. Clearly, this shows that both $\eta_{G,T}$ and $\eta_{M,T}$ can be only error indicators.

- Nevertheless you may obtain good results with this error indicators, see Chapter 3. In spite of being equivalent in the sense of the lemma the two indicators may produce very different results, see e.g. Table 11.

- The error indicators are problem independent.

- They work only on one element and do not need any communication in a parallel implementation.

2.3 Error Estimation with a Higher Order Recovery of the Gradient

This kind of error estimators is also known as Zienkiewicz - Zhu estimator. A description of it can be found e.g. in [BR93] or [Ver93], chapter I.5.

Let u_0 be the solution of our problem and u_h the current numerical approximation. One wants to estimate

$$\|\nabla u_0 - \nabla u_h\|_{L^p(T)}.$$

For this purpose we will construct an approximation U of ∇u_0 only using u_h. Then

$$\|U - \nabla u_h\|_{L^p(T)} \geq \Big| \|\nabla u_0 - U\|_{L^p(T)} - \|\nabla u_0 - \nabla u_h\|_{L^p(T)} \Big|$$

holds for each element T. If the approximation U is good enough, i.e.

$$\|\nabla u_0 - U\|_{L^p(T)} \ll \|\nabla u_0 - \nabla u_h\|_{L^p(T)},$$

then

$$\|U - \nabla u_h\|_{L^p(T)} \approx \|\nabla u_0 - \nabla u_h\|_{L^p(T)} = |u_0 - u_h|_{W^{1,p}(T)}$$

holds. Thus we have an approximation of the error in the $W^{1,p}$-seminorm and we set

$$\eta_{A,T;p} = \|U - \nabla u_h\|_{L^p(T)} \quad \forall T.$$

There are different possibilities to construct U. Generally the function U at a point is an average of the gradient in a neighborhood of that point.

In our implementation we assume that U is completely determined by its values in the nodes of the grid. If the grid consists of triangles we can identify U with a continuous and piecewise linear function in each component. Let A be a node of a element T. The value $U(A)$ is the arithmetical average of all gradients of the elements which have A as node.

In [Ver93] is proposed a different weighting. Let ω_A the union of all elements which have A as node. The gradient of element $T \subset \omega_A$ is weighted with $|T|/|\omega_A|$. This method has more memory requirements than the simple arithmetic average.

Remark 2.3

- This error estimator is problem independent.

- There is some local communication in a parallel implementation.

2.4 Estimating the Error with the Residual

Let F be an operator from the Banach space X to the dual of the Banach space Y with $F \in C^1(X, Y^*)$. We consider the problem $F(u) = 0$. Let u_0 be an isolated and regular solution. With some additional assumptions

$$0.5\|DF(u_0)\|^{-1}_{\mathcal{L}(X,Y^*)}\|F(u)\|_{Y^*} \leq \|u - u_0\|_X \leq 2\|DF(u_0)^{-1}\|_{\mathcal{L}(Y^*,X)}\|F(u)\|_{Y^*} \qquad (4)$$

holds in some neighbourhood of u_0. For more details and the proof we refer to [Ver93].

Inequality (4) says that we can estimate the error in the norm of the Banach space X with the norm of the residual in Y^*. Given a discrete solution u_h. Let us assume that it is in such a neighbourhood of u_0 that (4) holds. Then we want to estimate $\|F(u_h)\|_{Y^*}$. First, in general we cannot treat $F(\cdot)$ numerically. Therefore we discretize $F(\cdot)$ by projecting $F(\cdot)$ locally into a finite dimensional space. This defines an operator $\tilde{F}_h : X_h \mapsto Y^*$. Second, we cannot compute $\|\cdot\|_{Y^*}$. So we want to find a finite dimensional space $\tilde{Y}_h$ such that $\|\tilde{F}_h(u_h)\|_{\tilde{Y}_h^*}$ gives information about $\|F(u_h)\|_{Y^*}$. Under appropriate assumptions on $\|\tilde{F}_h(u_h)\|_{\tilde{Y}_h^*}$ the inequalities

$$\|F(u_h)\|_{Y^*} \leq c\|\tilde{F}_h(u_h)\|_{\tilde{Y}_h^*} + D, \quad c > 0,$$

and

$$\|F_h(u_h)\|_{\tilde{Y}_h^*} \leq C\|F(u_h)\|_{Y^*} + d, \quad C > 0.$$

hold. The terms D and d depend on the quality of the approximation of the differential operator, on the consistency error and the algebraic residual. These terms are in general of higher order. Using two dimensional linear elements the construction of $\tilde{Y}_h$ works with the help of the edge and element bubble functions. For more details we again refer to [Ver93].

In this way we obtain a problem dependent error estimator. Let T be an element, h_T its diameter (longest edge), E one edge of the element and h_E the length of this edge. Additionally, let n_E be the outward normal on the edge E. The jump of a function u across the edge E in the direction of n_E is denoted by $[u]_E$, i.e.

$$[u]_E(x) := \lim_{t \to +0} u(x + tn_E) - \lim_{t \to +0} u(x - tn_E) \quad \forall x \in E.$$

For our model problem the error estimator looks like

$$\eta_{R,T;p} = \left(h_T^p\| - \nabla_h \cdot \varepsilon\nabla_h u_h + b_h \cdot \nabla_h u_h - f_h\|^p_{0,p;T} + \sum_{E(T)\setminus\partial\Omega} h_E\|[n_E \cdot \varepsilon\nabla_h u_h]_E\|^p_{p;E} + \right.$$
$$\left. \sum_{E(T)\cap\partial\Omega_N} h_E \left\| \left(\varepsilon\frac{\partial u}{\partial n_E}\right)_h - n_E \cdot \varepsilon\nabla_h u_h \right\|^p_{p;E} \right)^{1/p}, 1 < p < \infty. \qquad (5)$$

Here $E(T)$ is the set of edges of T. If we have $u \in W^{1,p}(\Omega)$ (e.g. $u \in H^1(\Omega)$, $z \in H^{-1/2}(\Gamma_N)$ and $g \in H^{1/2}(\Gamma_D)$), then

$$\|u - u_h\|_{1,p} \leq c_1 \left(\sum_T \eta_{R,T;p}^p \right)^{1/p} + D, \quad c_1 > 0,$$

and

$$\eta_{R,T;p} \leq c_2 \|u - u_h\|_{1,p;\omega_T} + d \quad \forall T, \quad c_2 > 0,$$

hold ([Ver93] , proposition 12.1). The region ω_T includes T and all neighbour elements which have a common edge with T.

Next we introduce a modification of $\eta_{R,T;p}$. We omit the element residuals and obtain as an error estimator

$$\eta_{J,T;p} = \left(\sum_{E(T)\setminus\partial\Omega} h_E \|[n_E \cdot \varepsilon \nabla_h u_h]_E\|_{p;E}^p + \sum_{E(T)\cap\partial\Omega_N} h_E \left\| \left(\varepsilon \frac{\partial u}{\partial n_E} \right)_h - n_E \cdot \varepsilon \nabla_h u_h \right\|_{p;E}^p \right)^{1/p},$$

$1 < p < \infty$. In our numerical test we also used $p = 1$.

Remark 2.4 One can interpret $\eta_{R,T;p}$ as the residual with respect to the strong form of the equation (3). Then the second term of (5) reflects that u_h does not belong to $W^{2,p}$ for $p > 1$ and the third term gives the information how good u_h fulfills the discrete Neumann boundary conditions.

Remark 2.5 Suppose that the Dirichlet boundary condition has a jump, a situation like in Figure 1.

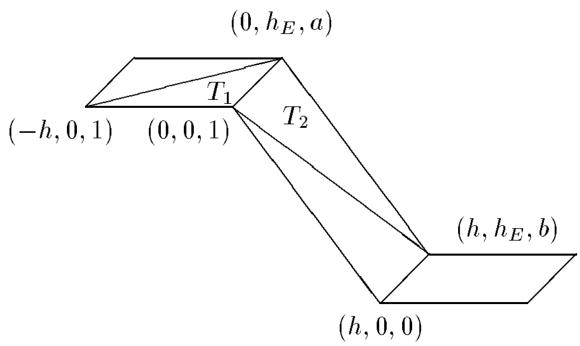

Figure 1: Jump with a linear triangulation

Let the boundary of T_1 be the x-axis and the height of the jump be 1. The vertices of T_1 has the coordinates $(0,0), (0,h_E), (-h,0)$. The neighbour element T_2 has the coordinates $(0,0), (h,h_E), (0,h_E)$ with the function values

$$u(0,0) = 1, \ u(-h,0) = 1, \ u(0,h_E) = a, \ u(h,h_E) = b.$$

We consider piecewise linear elements. Then

$$\nabla_{T_1} u = \begin{pmatrix} 0 \\ (a-1)/h \end{pmatrix}, \quad \nabla_{T_2} u = \begin{pmatrix} (b-a)/h \\ (a-1)/h \end{pmatrix}$$

holds. If we consider the jump of $\varepsilon\nabla u_h$ from T_1 to T_2 then the outward normal is

$$n_E = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Estimating the error with $\eta_{R,T;p}$ or $\eta_{J,T;p}$ we obtain

$$\begin{aligned} h_E \| n_E \cdot \varepsilon(\nabla_{T_1} u - \nabla_{T_2} u) \|^p_{L^p(E)} &= h_E \| \varepsilon(a-b)/h \|^p_{L^p(E)} \\ &= \frac{h_E^2}{h^p} \varepsilon^p |a-b|^p. \end{aligned}$$

If we have a regular triangulation then the ratio h_E/h must not tend to zero and the term $(a-b)$ tends to 1. So the error estimators $\eta_{J,T_1,p}$ respectively $\eta_{R,T_1,p}$ do not tend to zero for $h \to 0$ and $p \geq 2$, compare Example 3.2.

Remark 2.6 The error estimators $\eta_{A,T;p}$ on the one hand and $\eta_{R,T;p}$ respectively $\eta_{J,T;p}$ on the other hand are not equivalent on a single element. It is possible that $\eta_{A,T;p}$ indicates the local error zero whereas $\eta_{R,T;p}$ and $\eta_{J,T;p}$ do not.

An example is shown in Figure 2. Let the size of the elements be h and given a piecewise linear function u_h. Then u_h is determined by its values in the vertices of the triangles, the numbers near the vertices are the values of u_h. The gradient of u_h in T is $(0,0)^T$ whereas the gradient of the left neighbour is $(-1/h, 0)^T$. Thus $\eta_{J,T;p}$ and $\eta_{R,T;p}$ are greater than zero. We consider now the upper left vertex of T. Starting from T the gradients of the elements on this vertex are in counter clock direction $(0,0)^T$, $(1/h, 1/h)^T$, $(1/h, 0)^T$, $(0,0)^T$, $(-1/h, -1/h)^T$ and $(-1/h, 0)^T$. Thus the average gradient in this vertex is $(0,0)^T$. The same holds for the other vertices of T and $\eta_{A,T;p}$ yields zero.

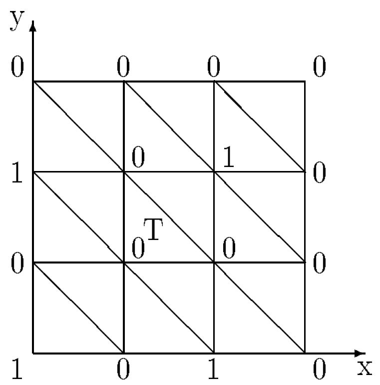

Figure 2: Nonequivalence of $\eta_{A,T;p}$ and $\eta_{J,T;p}$

2.5 Marking Methods

After having assigned each element a number we have to decide which elements will be refined and which not. This decision is called marking method and there exist several ones.

A widely used marking method stores during the evaluation of the error indication or estimation the maximum of the computed values values. We call it *maxerror*. Then an element T will be refined if $\eta_{.,T}$ satisfies

$$\eta_{.,T} \geq reftol * maxerror \,, \quad 0 < reftol < 1.$$

In future we refer to this strategy as $MAXLOCERR$.

If we use an error estimator we can search a set S as small as possible such that

$$\left(\sum_{T\in S}\eta_{.,T}^p\right)^{1/p} \le reftol \left(\sum_{T}\eta_{.,T}^p\right)^{1/p}, \quad 0 < reftol < 1. \tag{6}$$

The elements in S are responsible for a certain part of the global error. We refer to that strategy as $GLOBERR$. In practice we start like in $MAXLOCERR$ with a certain $reftol_0$. So we get a set S_0 which possibly does not satisfy (6). Then we decrease $reftol_0$ and repeat this until we have found a set S which satisfies (6).

Another strategy is to compute the estimated global error $(\sum \eta_{.,T}^p)^{1/p}$ and divide it by the number of elements. In this way an average error $\bar{\eta}_T$ is obtained and an element T is refined if

$$\eta_{.,T} \ge reftol * \bar{\eta}_T, \quad 0 < reftol.$$

A next strategy is to compare the error indication $\eta_{.,T}$ of an element with a constant and to refine this element if $\eta_{.,T}$ is greater than this constant. This makes sense only if such a constant can be chosen in a meaningful way.

If we choose only some elements for refinement in each step we can expect to receive a nearly optimal grid. But this strategy involves a lot of (possibly expensive) iteration steps in the solution process. Generally this will result in a large time to compute the solution. If we choose too much elements for refinement we get probably too much unknowns in our system but only need a few iteration steps. To achieve a nearly optimal time of the solution process we have to make a compromise.

In order to prevent the refinement of too few elements we demand a certain increase of the number of unknowns as result of the refinement. In practice we demand that a certain percentage of elements, which may be refined, will be refined. If with the strategy $MAXLOCERR$ and a given $reftol$ we do not achieve this percentage the program will automatically decrease $reftol$ and then mark once again. Using the strategy $GLOBERR$ the set S is increased as long as both (6) is fulfilled and the prescribed percentage of marked elements is obtained.

In this paper we only state some results with two marking methods. We will pay attention on a comparison of more marking methods in future.

3 Numerical Experiments

We carried out our numerical tests on a MIMD transputer system PARSYTEC GCel/128 with T 805. This system is known for having a very fast communication compared with the performance of the processors.

If not stated otherwise we used 64 processors for the computations. On each processor are 4 MB memory available. After loading the code we could use roughly 2.5 MB. The error estimators are implemented as part of the parallel version of the software package *ug* whose main parts were developed by P. Bastian, for some details see [Bas93].

3.1 Comparing Error Estimators with Respect to the Accuracy

As criterion of comparison we use the accuracy in some norm of the discrete solution. In the following tables we present the results of a set of tests. The parameters of some tests are presented in Table 1. These tests show the behaviour of the indicators and estimators, respectively, applied with non special parameters (as an user without a-priori knowledge on the examples and adaptive techniques would do). Surely there are better and worse sets of parameters for each estimator and each problem. But it cannot be the sense of adaptive techniques at first to test a lot of sets of parameters in order to find a set which achieves a good result. Rather there should be a wide range of parameters which produce reasonable results. In this range the dependence of the results on the parameters should be weak. The parameters in our tests are

baselevel	the level up to which the mesh is uniformely refined
maxlevel	the maximal refinement level
mark method	the marking method
reftol	the refinement tolerance for the marking methods
min.ref.	the minimal percentage of elements that must be refined relative to the number of elements which are allowed to be refined

Our final aim is an adaptive choice of these parameters.

We use the following procedure

STEP : solve $\longrightarrow$ estimate $\longrightarrow$ load balance $\longrightarrow$ refine

In all test we used linear triangular elements. We terminated the solution process if something went wrong for insufficient memory. Because the strong limitation of memory this happened often rather quickly. The reason is that the processors need a short time the triple memory for doing the load balance.

The l_∞-errors given in the tables below are the maximal difference $|u_0(x_i) - u_h(x_i)|$ where u_0 is the solution, u_h the current numerical approximation and x_i is a node of the grid. In some examples we restricted the set of nodes which we choose for the comparison. The error estimator *exact* is an approximation of the L^2-error or the error in the $W^{1,2}$-seminorm. Intentionally we often choose a low *baselevel* in order to let the adaptive strategy do the main work of building the grid.

Table	baselevel	maxlevel	mark method	reftol	min. ref.
2	3	9	MAXLOCERR	0.5	0.125
3	3	9	MAXLOCERR	0.5	0.15
4	3	9	GLOBERR	0.5	0.125
5	2	10	MAXLOCERR	0.5	0.125
6	2	10	MAXLOCERR	0.25	0.1
7	2	10	GLOBERR	0.75	0.2
8	2	10	MAXLOCERR	0.25	0.125
9	4	8	MAXLOCERR	0.25	0.125

Table 1: Parameters of the presented tests

For the most examples we used as discretization finite elements with Samarskij upwind, see e.g. [Tur91]. For Example 3.1 we did not use upwinding and for Example 3.4 we used finite volumes with upwinding.

Example 3.1 This is a Dirichlet's problem for the Laplace operator :

$$\begin{aligned} -\Delta u &= f(x,y) \quad \Omega=(0,1)^2 \\ g(x,y) &= \begin{cases} 1/(1+e^{-200(y-0.8)}) & x=0 \\ 1/(1+e^{-200(x-0.8)}) & y=0 \\ 1/(1+e^{-200(\sqrt{1+y^2}-0.8)}) & x=1 \\ 1/(1+e^{-200(\sqrt{1+x^2}-0.8)}) & y=1 \end{cases} . \end{aligned}$$

The right hand side is chosen such that

$$u(x,y)=\frac{1}{1+e^{-200(\sqrt{x^2+y^2}-0.8)}},$$

is the exact solution, see Figure 3, i.e.

$$f(x,y)=-\frac{e^{200\sqrt{x^2+y^2}+160}\left[e^{160}(200\sqrt{x^2+y^2}+1)+e^{200}\sqrt{x^2+y^2}(1-200\sqrt{x^2+y^2})\right]}{\left(e^{160}+e^{200\sqrt{x^2+y^2}}\right)^3\sqrt{x^2+y^2}}.$$

The Dirichlet boundary condition is continuous but has almost jumps at the points $(0,0.8)$ and $(0.8,0)$.

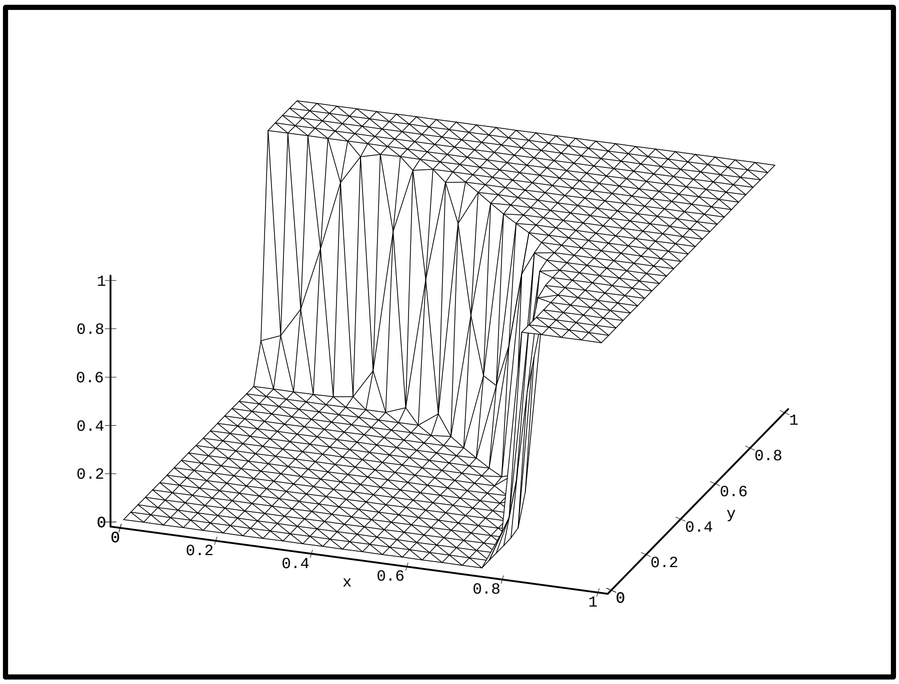

Figure 3: Solution of Example 3.1

RESULTS : An uniform refinement seems to be the best in this example. It is interesting to compare the results of this example with the results of Example 3.3, Table 5. The shape of the solutions are similar (only the direction of the jump is different) but in Example 3.3 uniform refinement works much worse than all adaptive strategies. The results of all error indicators and estimators are similar.

step		uniform	L^2-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$
4	nodes	1089	452	427	469	419	419	441
	l_∞-error	3.065	3.253	3.166	3.191	2.865	2.865	3.049
	$W^{1,2}$-error	7.572	9.483	7.656	7.712	3.056	3.056	7.533
7	nodes	66049	1912	2504	2396	2213	2213	2856
	l_∞-error	0.005	1.254	0.099	0.205	0.069	0.069	0.116
	$W^{1,2}$-error		2.467	0.602	0.749	0.525	0.525	0.407
10	nodes		9891	12322	22939	13778	13778	18534
	l_∞-error		0.064	0.089	0.090	0.059	0.059	0.070
	$W^{1,2}$-error		0.360	0.427	0.386	0.180	0.180	0.179
	residual		2.9-4	8.1-5	1.7-4	1.0-4	1.0-4	1.3-4
last	nodes		56413	61684	70792	77600	77600	69403
	l_∞-error		0.049	0.128	0.084	0.054	0.054	0.072
	$W^{1,2}$-error			0.119	0.167	0.125	0.125	0.139
	residual		2.9-5	9.9-5	8.1-5	1.1-4	1.1-4	1.3-4
last		7	14	15	15	14	14	13

Table 2: Example 3.1

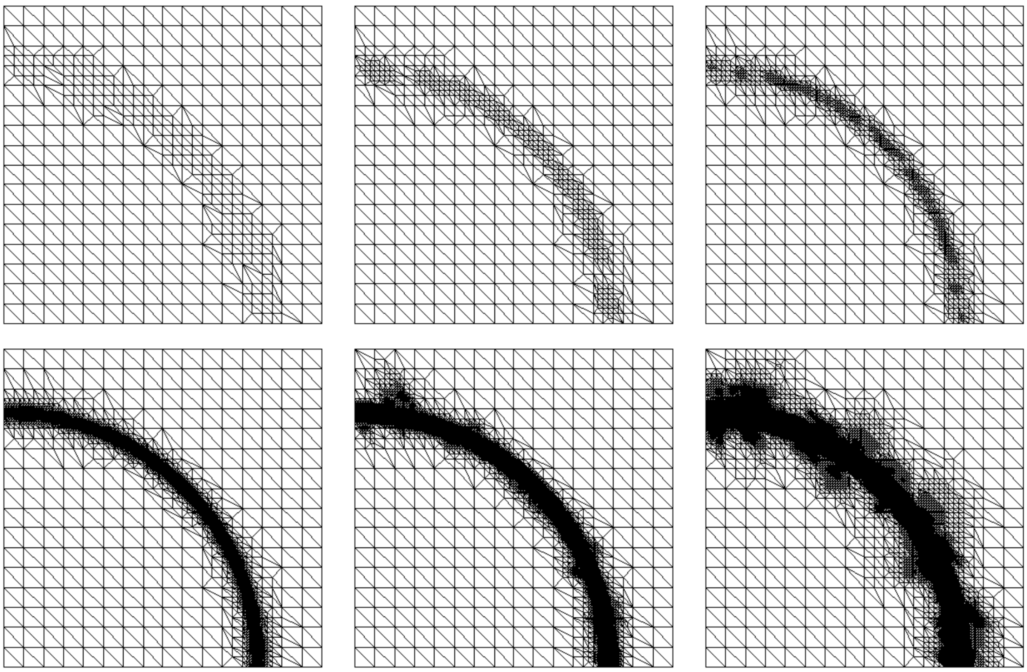

Figure 4: Development of the mesh, Example 3.1, $\eta_{J,T;2}$, Table 2, step 4,5,6,8,10 and 12

Example 3.2 This example is a convection diffusion-equation with a constant convection vector and jumps in the Dirichlet boundary condition:

$$\begin{array}{rcl} -\varepsilon\Delta u + (1,0)\cdot\nabla u &=& 0 \quad \Omega = (0,1)^2 \\ g(x,y) &=& 0 \quad \text{for} \quad \left\{\begin{array}{lll} y = 0 & \text{and} & x \le 1 \\ y = 1 & \text{and} & x \le 1 \\ x = 0 & \text{and} & |y-0.5| > 0.05 \end{array}\right. . \\ g(x,y) &=& 1 \quad \text{otherwise} \\ \varepsilon\frac{\partial u}{\partial n} &=& 0 \quad \text{on } \{x = 1 \text{ and } y \le 1\} \end{array} \tag{7}$$

The solution of (7) is

$$u(x,y) = \sum_{k=1}^{\infty} \phi_k(x)\psi_k(y) = \sum_{k=1}^{\infty}(\alpha_k\phi_{1k}(x) + \beta_k\phi_{2k}(x))\psi_k(y),$$

where

$$\phi_{1k}(x) = e^{x/(2\varepsilon)}\frac{\sinh(b_k(1-x))}{\sinh(b_k)}, \quad \phi_{2k}(x) = e^{(x-1)/(2\varepsilon)}\frac{\sinh(b_k x)}{\sinh(b_k)}$$

with

$$b_k = \sqrt{\frac{1}{(2\varepsilon)^2} + (k\pi)^2}.$$

The constants are

$$\alpha_k = \frac{2}{k\pi}(\cos(0.45k\pi) - \cos(0.55k\pi)), \quad \beta_k = \alpha_k\left(\frac{b_k e^{1/(2\varepsilon)}}{\sinh(b_k)}\right) / \left(\frac{1}{2\varepsilon} + b_k\frac{\cosh(b_k)}{\sinh(b_k)}\right)$$

and the functions only depending on y are

$$\psi_k(y) = \sin(k\pi y).$$

We carried out our tests with $\varepsilon = 10^{-3}$. For a graphical representation of the solution see Figure 5.

Results : The numerical solution of this example tends to a constant l_∞-error in a neighbourhood of the jumps of the Dirichlet boundary condition. The constant depends on the maximal refinement level and is independent of the error estimator. Therefore we did not take into consideration the error in a neighborhood of this points in Tables 3 and 4. More precisely we only calculated the error for points with $x \ge 0.5$. It turned out that then the maximal error often occurred in the point $(1, 0.5)$. The solution of this problem does not belong to $W^{1,2}(\Omega)$. For the numerical tests we defined a piecewise linear function with the same values in the nodes as the solution. Then we used this function in order to estimate the error in some norm. The results for the L^2-norm and the $W^{1,2}$-seminorm are presented in columns 'L^2-ex.' and '$W^{1,2}$-ex.', respectively.

The different working of $\eta_{M,T}$ and $\eta_{J,T;2}$ can be seen in Figure 6. From Table 3 we see that the l_∞-error at the end of the solution process is smaller for $\eta_{J,T;2}$. But the value of the point $(1, 0.5)$ is better for $\eta_{M,T}$. The middle part of the solution ($y \approx 0.5$) is resolved better by $\eta_{M,T}$. This error indicator generates a coarse grid where the solution is relatively constant and in this region the maximal error occurs. The lower and upper contour line are much worse than for $\eta_{J,T;2}$.

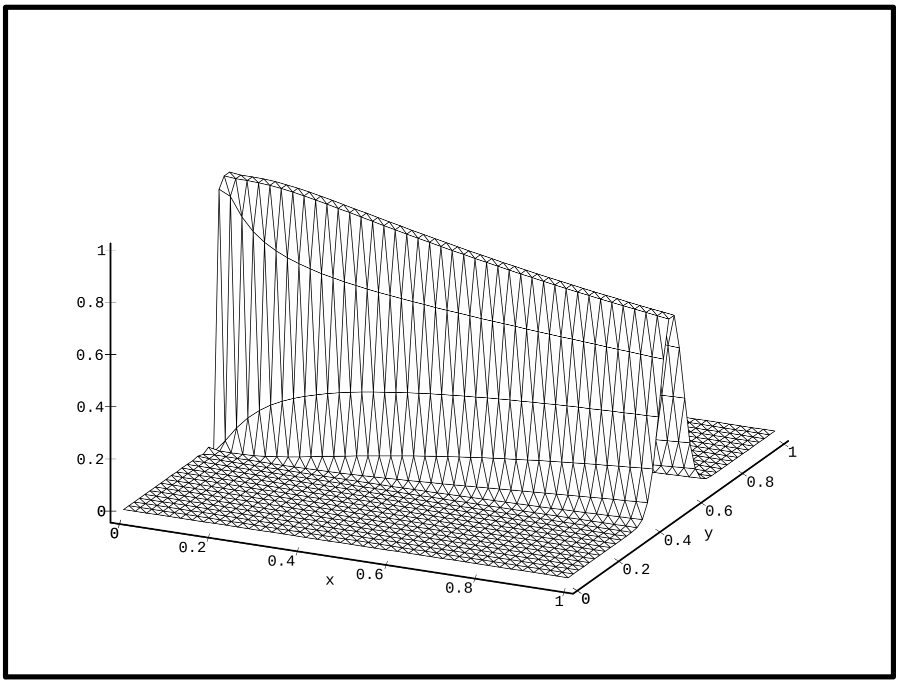

Figure 5: Solution of Example 3.2

In Figure 8 we see a grid development using $\eta_{J,T;2}$. With $\eta_{J,T;2}$ we actually estimate the error in the $W^{1,2}(\Omega)$-norm. But we know that the solution does not belong to $W^{1,2}(\Omega)$. Therefore, the error at the points in which the Dirichlet boundary conditions have a jump does not tend to zero. So we can observe that in a neighbourhood of these points a very deep refinement starts until some parameters (namely $MAXLEVEL$ and $MINREF$) stop this process and the refinement works well. These parameters compensate the wrong norm. If we switch these parameters off, the result looks like in Figure 7.

step		uniform	L^2-ex.	$W^{1,2}$-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$
4	nodes	1089	435	439	427	598	469	469	444
	l_∞-error	0.541	0.617	0.617	0.568	0.590	0.542	0.542	0.552
7	nodes	66049	7062	3788	5047	5189	3600	3600	3006
	l_∞-error	0.112	0.207	0.239	0.306	0.293	0.303	0.303	0.325
10	nodes			49021	40975	81818	47066	47066	21768
	l_∞-error			0.050	0.092	0.064	0.061	0.061	0.106
last	nodes		49303		100714		87664	87665	92709
	l_∞-error		0.073		0.103		0.031	0.031	0.033
last		7	9	10	11	10	11	11	12

Table 3: Example 3.2

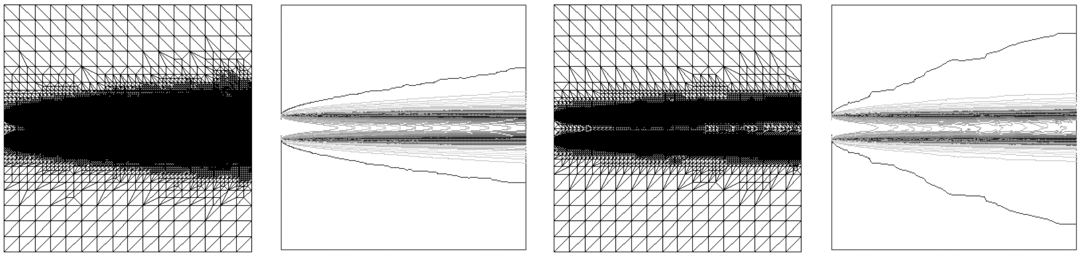

Figure 6: A comparison of the results of $\eta_{J,T;2}$ on the left hand side and $\eta_{M,T}$ on the right hand side

step		L^2-ex.	$W^{1,2}$-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$
4	nodes	435	439	451	433	404	404	455
	l_∞-error	0.617	0.617	0.567	0.567	0.572	0.572	0.543
7	nodes	7062	3788	2576	1974	2071	2071	2823
	l_∞-error	0.207	0.239	0.421	0.436	0.432	0.432	0.337
10	nodes	80061	26792	41271	30724	13808	13808	14667
	l_∞-error	0.049	0.087	0.128	0.145	0.140	0.140	0.140
last	nodes		54440	52041	39620	57230	57230	55171
	l_∞-error		0.050	0.079	0.118	0.047	0.047	0.052
last		10	11	11	11	12	12	12

Table 4: Example 3.2

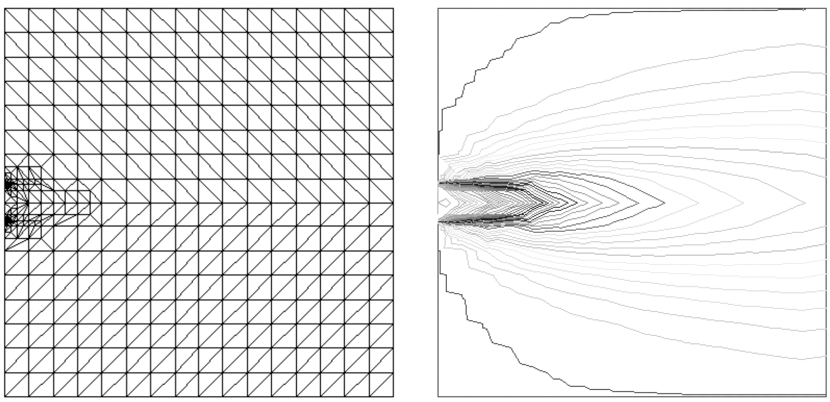

Figure 7: Using $\eta_{J,T;2}$ without compensating the wrong norm, step 20

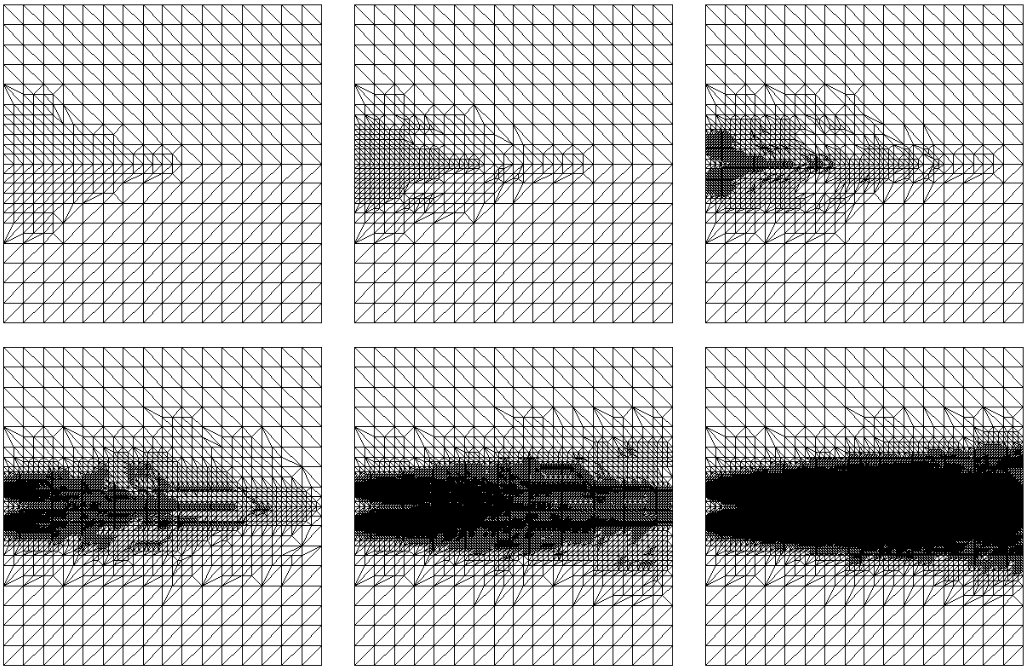

Figure 8: Development of the mesh, Example 3.2, $\eta_{J,T;2}$, Table 4, step 4,5,6,8,10 and 12

Example 3.3 This example contains a convection-diffusion equation with a nonconstant convection vector and a jump in the Dirichlet boundary conditions:

$$\begin{array}{rcll} -\varepsilon\Delta u + (y,-x)\cdot\nabla u &=& 0 & \Omega = (0,1)^2 \\ g(x,y) &=& 0 & \text{for } \left\{ \begin{array}{ll} y = 1 & \text{and } x \le 1 \\ x = 0 & \text{and } y > 0.7 \end{array} \right. \\ g(x,y) &=& 1 & \text{otherwise} \\ \varepsilon\frac{\partial u}{\partial n} &=& 0 & \text{on } \{y = 0 \text{ and } x \le 1,\ x = 1 \text{ and } y \le 1\} \end{array} .$$

The weak solution tends to

$$u = \left\{ \begin{array}{ll} 1 & \sqrt{x^2+y^2} < 0.7 \\ 0 & \sqrt{x^2+y^2} > 0.7 \end{array} \right.$$

for $\varepsilon \to +0$. In the numerical tests we used $\varepsilon = 10^{-4}$.

Results : In this example there is also the problem with the jump in the Dirichlet boundary conditions. The l_∞-error given in Table 5 is restricted to the nodes where $\left|\sqrt{x^2+y^2} - 0.7\right| \ge 0.025$.

The error estimators $\eta_{J,T;2}$ and $\eta_{R,T;2}$ refine the grid in such a way that the amount of memory grows unfavourable to the available memory. The early termination of the solution process causes a larger error than with other estimators. The simple error indicators worked well.

step		uniform	L^2-ex.	$W^{1,2}$-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$
3	nodes	289	114	125	125	122	147	147	162
	l_∞-error	0.461	0.687	0.527	0.612	0.612	0.490	0.490	0.553
7	nodes	66049	687	1188	1550	3717	1157	1157	1485
	l_∞-error	0.250	0.377	0.333	0.423	0.331	0.415	0.415	0.409
11	nodes		16753	20727	39504		9291	9291	11682
	l_∞-error		0.167	0.134	0.080		0.265	0.265	0.253
last	nodes		102802		60297	63900	42561	42561	66543
	l_∞-error		0.056		0.058	0.077	0.120	0.120	0.077
last		7	13	11	12	10	14	14	15

Table 5: Example 3.3

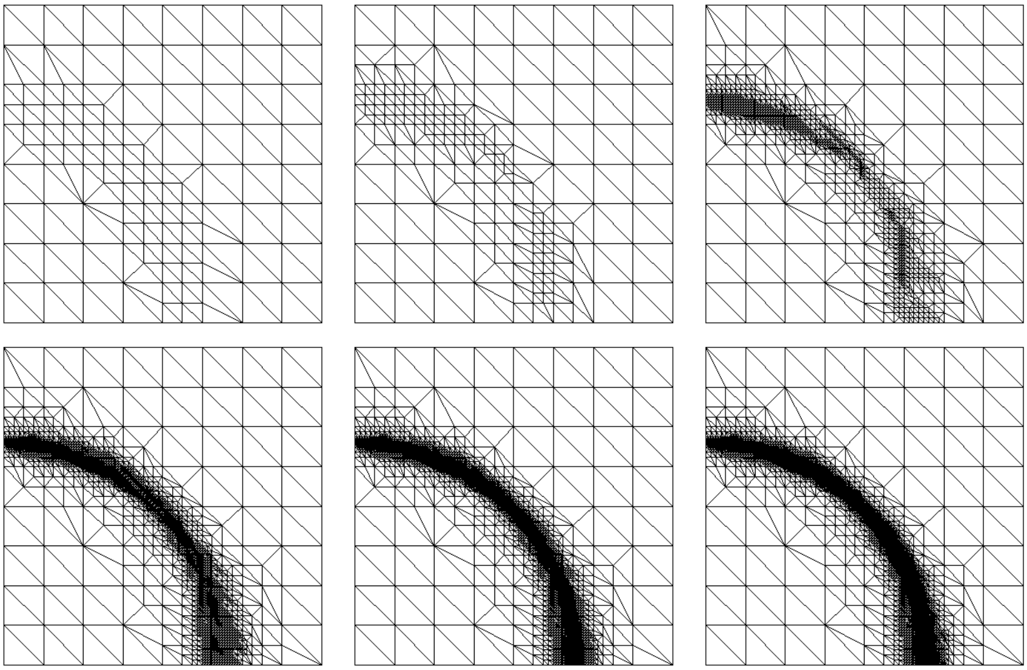

Figure 9: Development of the mesh, Example 3.3, $\eta_{M,T}$, Table 5, step 3,4,6,8,10 and 12

Example 3.4 Let e_ρ and e_ϕ be the usual unit basis vectors of a polar coordinate system. We consider the following problem on a circular ring:

$$\begin{array}{rcll} -\varepsilon\Delta u + e_\rho \cdot \nabla u &=& 0 & \Omega = \{(x,y) : 0.25 \le x^2+y^2 \le 1\} \\ g(x,y) &=& \sin(\phi) & \text{for } r=1, \\ g(x,y) &=& \frac{e^{-0.75/\varepsilon}}{0.25}\sin(\phi) & \text{for } r = 0.25 \end{array} . \tag{8}$$

where $\phi = \arg((x,y))$. The solution of (8) is

$$u(r,\phi) = \frac{e^{(r-1)/\varepsilon}}{r}\sin(\phi).$$

In the tests we set $\varepsilon = 10^{-3}$. The solution for this parameter is represented in Figure 10.

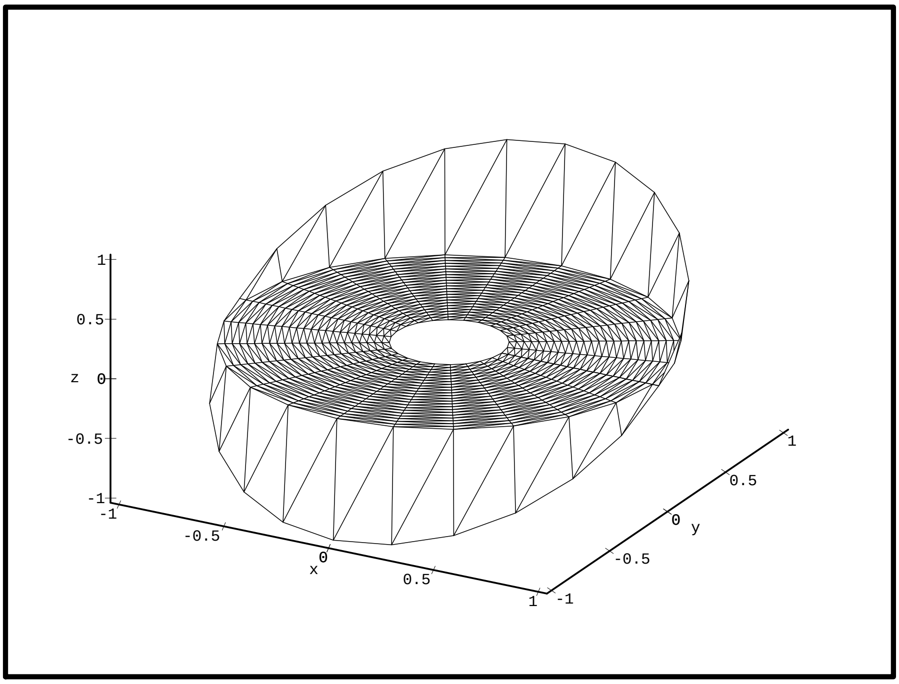

Figure 10: Solution of Example 3.4

Results : The l_∞-error given in Tables 6 and 7 is the maximal difference of the computed and the real value in the nodes. Due to the piecewise polynomial approximation of the boundary and the steep gradient of the solution at the boundary there is a constant l_∞-error in the nodes near the boundary depending on the maximal fineness of the mesh. Therefore we considered only such nodes whose distance from the origin is less than 0.99. The criterion of termination was the maximal number of 12 steps in Table 6 and 11 steps in Table 7 (or memory overflow). So there are after having reached the maximal level one or two post-iterations.

In this example we got the worst results when we used as estimator the exact error in L^2-norm. The situation with the error in the $W^{1,2}$-seminorm is better but the results are still worse than, for instance, the results received with the simple error indicators. The results with $\eta_{R,T;2}$ and $\eta_{J,T;2}$ with this strategy are poor, too. We got better if we had weaken the norm in $\eta_{J,T;p}$ to $p = 1$. The simple error indicators $\eta_{M,T}$ and $\eta_{G,T}$ produced good results in our tests. For a graphical representation of the resulting meshes due to Table 6 see Figure 11. The critical regions seem to be the neighbourhood of $(-1, 0)$ and $(1, 0)$, respectively. The refinement of this regions seems to be the reason for the better behaviour of $\eta_{M,T}$ or $\eta_{J,T;1}$. The post-iteration steps influence the l_∞-error in the nodes and in some tests it decreases considerably.

step		unif.	L^2-ex.	$W^{1,2}$-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$	$\eta_{J,T;1}$
3	nodes	432	272	293	226	240	274	274	292	270
	l_∞-error	0.071	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013
6	nodes	24960	1944	2676	1744	1702	1202	1202	2538	1106
	l_∞-error	0.094	0.199	0.094	0.094	0.095	0.240	0.240	0.094	0.053
10	nodes		19402	42395	33496	34076	8014	8014	44610	9654
	l_∞-error		0.256	0.031	0.020	0.020	0.072	0.072	0.014	0.031
	residual		7.1-6	1.3-6	2.6-6	1.5-6	4.9-6	4.9-6	9.7-7	1.7-6
last	nodes	99072	35879		44639	48614	19672	19672		24164
(12)	l_∞-error	0.027	0.269		0.012	0.012	0.071	0.071		0.005
	residual	2.7-6	1.8-6		7.7-7	1.1-6	1.3-6	1.3-6		1.2-6
last		7	12	10	12	12	12	12	10	12

Table 6: Example 3.4

step		L^2-ex.	$W^{1,2}$-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$	$\eta_{J,T;1}$
3	nodes	248	256	226	210	218	218	284	292
	l_∞-error	0.022	0.022	0.013	0.013	0.013	0.013	0.013	0.013
6	nodes	2289	2260	1600	1552	1600	1600	2264	2450
	l_∞-error	0.116	0.094	0.095	0.150	0.095	0.095	0.115	0.248
10	nodes	33203	32556	23896	27026	32124	32124	27626	34191
	l_∞-error	0.157	0.075	0.030	0.029	0.030	0.030	0.023	0.006
	residual	1.8-6	1.3-6	2.9-6	1.4-6	2.0-6	2.0-6	1.0-6	1.1-6
11	nodes	42999	46254	35590	36548	42706	42706	43626	55994
	l_∞-error	0.160	0.116	0.024	0.023	0.024	0.024	0.018	0.002
	residual	3.1-6	3.8-6	1.2-6	9.3-7	2.4-6	2.4-6	4.9-7	2.3-6

Table 7: Example 3.4

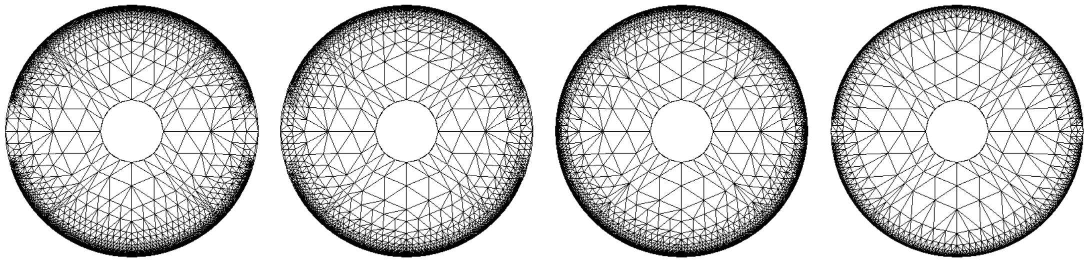

Figure 11: Results of different error estimators on Example 3.4 after step 12, from left to right $L^2 - exact$, $\eta_{J,T;2}$, $\eta_{J,T;1}$, $\eta_{M,T}$

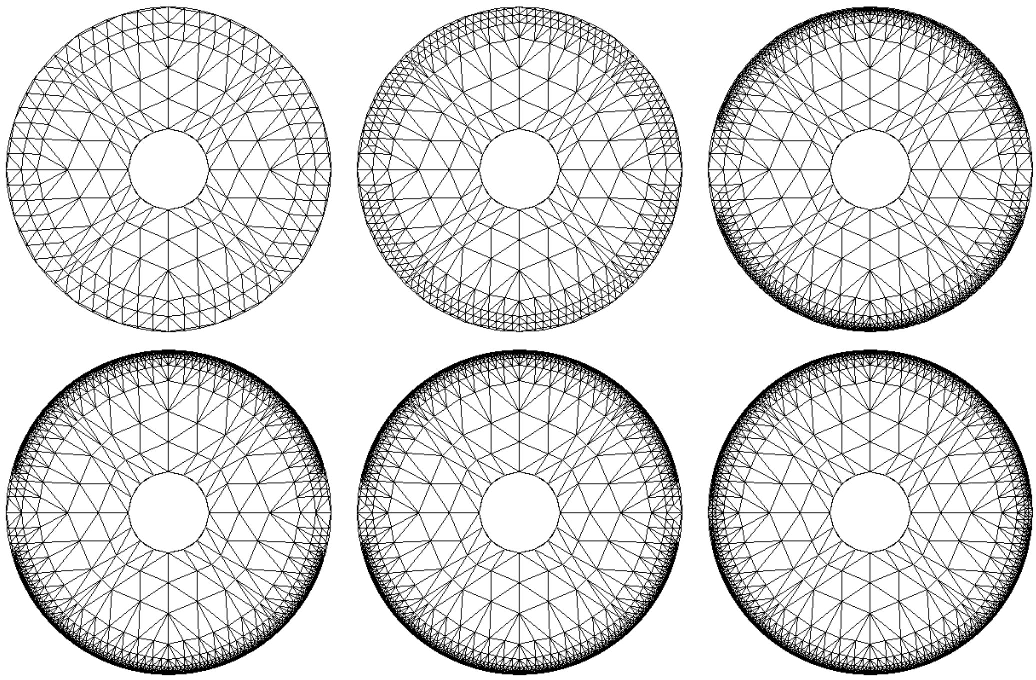

Figure 12: Development of the mesh, Example 3.4, $\eta_{G,T}$, Table 6, step 3,4,6,8,10 and 12

Example 3.5 Last we investigate an example with more than one layer. The layers are of exponential and polynomial form:

$$-10^{-2}\Delta u + \left(1 + \frac{1}{100(x-1.01)}, \frac{1}{10}\left(1 + \frac{1}{10(y-1.1)}\right)\right) \cdot \nabla u =$$

$$-450\left(1 + \frac{1}{100(x-1.01)}\right)(1-x)^9 - \frac{81}{2}(1-x)^8 \quad \Omega = (0,1)^2$$

$$g(x,y) = \begin{cases} e^{100(x-1)}(x-1.02)(-1.2) + 45(1-x)^{10} & y = 0 \\ e^{10y}(-0.02)(y-1.2) & x = 1 \\ e^{100(x-1)+10}(x-1.02)(-0.2) + 45(1-x)^{10} & y = 1 \\ e^{-100+10y}(-1.02)(y-1.2) + 45 & x = 0 \end{cases}.$$

The solution is

$$u(x,y) = e^{100(x-1)+10y}(x-1.02)(y-1.2) + 45(1-x)^{10}$$

and it is represented in Figure 13.

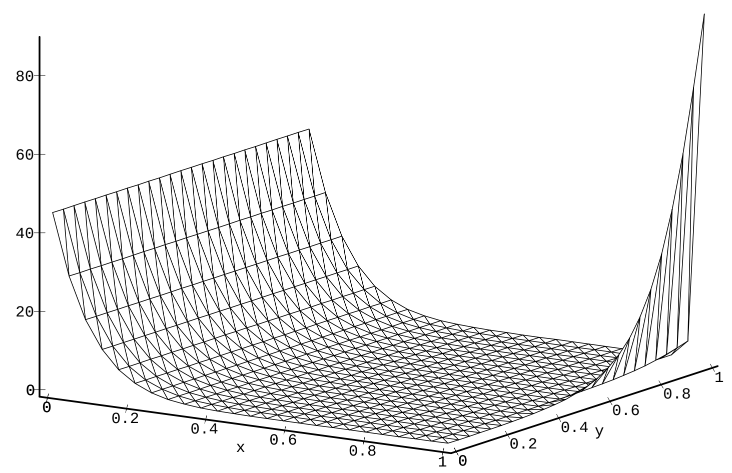

Figure 13: Solution of Example 3.5

RESULTS : At first we discuss Table 8. Regarding the errors of the results, an uniform refinement produces the best result although the reached refinement level is less than with all error indicators and estimators. In the results of the indicators and estimators are great differences, too. All find the regions with a great error but in this example it can be seen that this is not enough. A critical subregion seems to be the region at the boundary $x = 1$ for small y. Estimators refining in this region obtained better results than others, see Figure 14.

With the a-priori information of Table 8 it seems more reasonable to refine uniformly up to a higher level than 2 and to decrease the maximal level. The results of a test set are shown in Table 9. There the results are clearly better than in Table 8. The estimators $\eta_{R,T;2}$, $\eta_{J,T;2}$ and $\eta_{A,T;2}$ produce results with slightly less errors and with fewer unknowns than the indicators $\eta_{M,T}$ and $\eta_{G,T}$.

step		uniform	$W^{1,2}$-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$
3	nodes	289	219	260	159	118	118	183
	l_∞-error	8.833	8.588	8.751	7.473	19.195	19.195	9.003
	$W^{1,2}$-error	39.462	40.544	39.078	37.506	69.656	69.856	39.606
7	nodes	66049	5328	9417	2219	1252	1252	3287
	l_∞-error	0.086	1.699	1.069	5.763	2.407	2.407	0.663
	$W^{1,2}$-error	0.599	7.467	3.097	20.967	13.081	13.081	4.795
10	nodes		30161	54533	40692	10782	10782	34965
	l_∞-error		0.598	0.488	6.292	2.146	2.146	0.343
	$W^{1,2}$-error		2.500	1.996	16.715	4.237	4.237	1.081
	residual		3.2-5	1.3-5	8.3-5	1.9-5	1.9-5	1.1-5
11	nodes		52777		91685	22592	22592	65420
	l_∞-error		0.527		6.350	2.523	2.523	0.522
	$W^{1,2}$-error		1.946		16.200	4.162	4.162	0.883
	residual		3.4-5		6.3-5	4.1-5	4.1-5	4.9-6

Table 8: Example 3.5, $\eta_{M,T}$ 9 steps

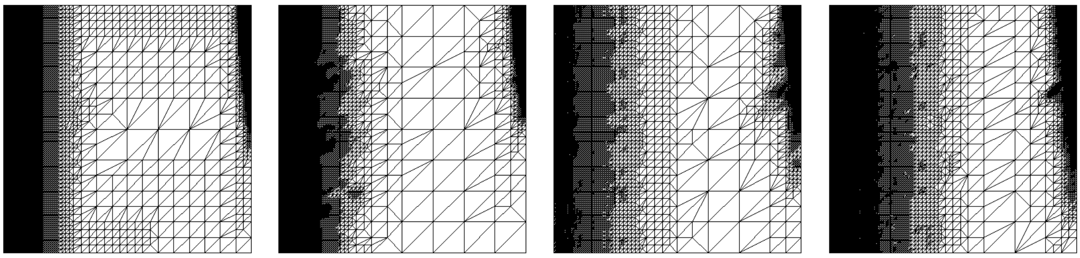

Figure 14: Results of different error estimators on Example 3.5 after step 11, from left to right $\eta_{M,T}$, $\eta_{G,T}$, $\eta_{J,T;2}$, $\eta_{A,T;2}$

step		$W^{1,2}$-ex.	$\eta_{M,T}$	$\eta_{G,T}$	$\eta_{R,T;2}$	$\eta_{J,T;2}$	$\eta_{A,T;2}$
5	nodes	2377	1796	1699	1725	1725	1899
	l_∞-error	1.048	1.180	1.196	1.194	1.194	1.023
	$W^{1,2}$-error	7.627	6.675	6.852	8.246	8.246	6.681
7	nodes	8533	9756	5978	5485	11542	6967
	l_∞-error	0.424	0.335	0.518	0.278	0.278	0.147
	$W^{1,2}$-error	3.780	2.310	3.372	2.327	2.327	1.889
8	nodes	18227	25184	8448	8020	8020	10862
	l_∞-error	0.380	0.338	0.523	0.286	0.286	0.123
	$W^{1,2}$-error	2.738	1.693	3.169	2.140	2.140	1.568
	residual	2.9-5	8.0-6	8.5-6	7.3-6	7.3-6	7.0-6
10	nodes	48620	62694	44276	40331	40331	33541
	l_∞-error	0.181	0.274	0.282	0.040	0.040	0.040
	$W^{1,2}$-error	0.776	0.589	0.882	0.438	0.438	0.557
	residual	8.8-6	3.7-6	7.2-6	2.6-6	2.6-6	2.2-6

Table 9: Example 3.5

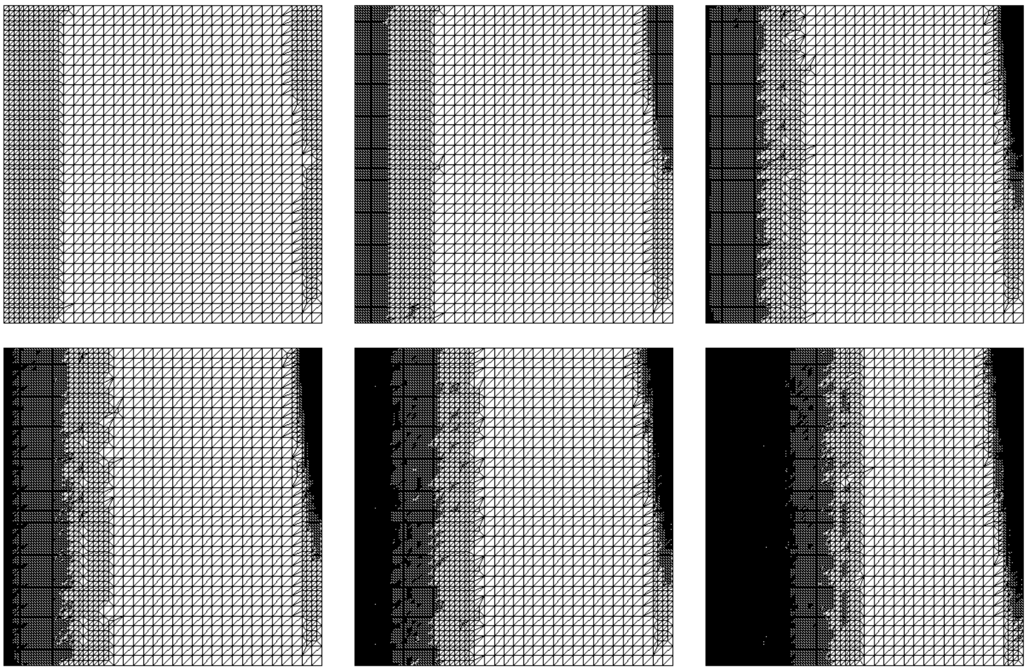

Figure 15: Development of the mesh, Example 3.5, $\eta_{A,T;2}$, Table 9, step 5,6,7,8,9 and 11

3.2 The Parallel Behaviour

The algorithms for estimating the error in a serial code and in a parallel code look different. The algorithm in our *serial* code has the following form:

1. go through the list of elements
   - compute element gradients etc.
2. go through the list of elements
   - receive information of all neighbour elements
   - compute the local error estimate

In contrast to the serial algorithm the *parallel* one looks as follows:

1. go through the list of elements
   - compute element gradients etc.
2. go through the list of elements
   - receive information of the neighbour elements which are on the same processor
   - store the received information from the neighbours, this step may result in a larger requirement of memory per element than in the serial algorithm
3. communicate in order to receive the information from foreign neighbour elements
4. go through the list of elements
   - compute the local error estimate

One criterion for the comparison of error estimators is the time necessary for estimating the error. This time should be small compared to the total solution time. Of course this time depends on the number of unknowns or elements and in a parallel code on the load balance. The assembling time also depends on the same features (and on the discretization). That's why we compare the time for doing the estimate with the assembling time. We demand that the estimation time should be less than the assembling time.

We also investigate the behaviour of the error estimators and indicators relating to the number of processors. The computation of $\eta_{M,T}$ and $\eta_{G,T}$ does not need any communication and the computation of $\eta_{R,T;p}$, $\eta_{J,T;p}$ and $\eta_{A,T;p}$ need communication only between neighbouring elements. Independent of the indicator or estimator we have to compute the global maximal local error or the global error. This depends on the marking method (actually we compute both). This step involves of course some communication, but only one time before starting the marking process. One can define a parallel efficiency index between m and n processors $(m < n)$ as follows

$$\mathrm{eff}(m,n) = \frac{m\ t(m)}{n\ t(n)}.$$

Since we demand the smallness of the estimation time in comparison with the total solution time this efficiency index is in practice not very important.

Some results with Example 3.3 are shown in Table 10. The parameters are the same as in Table 5, the first column are the times for step 7 and the second for step 11 (respectively 10 for $\eta_{G,T}$). In Table 11 we present results with Example 3.4 according to Table 6.

- The estimation time is in fact really small compared to the total solution time, which is generally some minutes.

- In our implementation the estimating process is clearly faster than the assembling process (both FEM and FVM). But note, if we compute estimates in some L^p-norm with $p \notin \{1, 2\}$ we have to use routines which need more time and the comparison will be worse.

- For sufficient load on the processors (enough elements or unknowns) the ratio between estimation time and assembling time is nearly constant. These constants give a criterion for the speed of the different indicators and estimators which is independent of the number of unknowns. We see that the error estimators roughly need twice the time for doing the estimate compared to the error indicators.

- For a large enough load the efficiency index lies between 0.7 and 0.9.

- For small loads on the processors the efficiency index may drop to 0.1.

The results presented in Tables 10 and 11 depend on the current implementation and may be changed in the next implementation. They should only provide an idea of the order of magnitude of the used time.

For an efficient implementation of the error estimators with respect to the computing time we need a certain amount of memory per element. For instance we want to avoid a multiple calculations of element gradients. Some error estimators need in a serial implementation less memory per element than in the parallel implementation (in particular $\eta_{A,T;p}$). From our point of view is the allocation of memory per element which is used *only* for the error estimator a waste of memory. Therefore we restricted ourselves to a certain amount of memory per element which we want to spend. The presented results should be considered under this restriction.

4 Concluding Remarks

- In the presented examples no error indicator or estimator is in general superior or inferior compared to the others.

- With respect to the accuracy of the solution the error estimators often worked slightly better than the error indicators. But the estimators need more than twice of the computing time to estimate the error. On the other hand we have seen that in both cases the computing time is very small compared to the total solution time.

- There are examples where a refinement of the elements with the greatest local error is not sufficient for obtaining a reasonable decrease of the global error. That's why sometimes even a very exact error estimation does not seem reasonable. In general it seems to be better to accept a certain over-refinement.

- The uniform refinement should not terminate too early. That seems to be connected with the so-called 'saturation assumption' in the theory of error estimators, which says that the discrete solution must be in some sense 'good enough' in order that the theory can be applied.

proc		$\eta_{M,T}$		$\eta_{G,T}$		$\eta_{R,T;2}$		$\eta_{J,T;2}$		$\eta_{A,T;2}$	
1	est.	0.70				1.50		0.96		1.57	
	ass.	3.44				2.61		2.61		3.78	
	ratio	0.20				0.57		0.37		0.42	
4	est.	0.22		0.52		0.51		0.33		0.56	
	ass.	1.00		2.17		0.77		0.77		1.29	
	ratio	0.22		0.24		0.66		0.43		0.43	
16	est.	0.10		0.17		0.19	1.35	0.12	0.84	0.17	0.99
	ass.	0.36		0.65		0.22	2.23	0.22	2.23	0.35	2.38
	ratio	0.28		0.26		0.86	0.61	0.55	0.38	0.49	0.42
64	est.	0.04	0.35	0.06	0.74	0.09	0.38	0.06	0.24	0.06	0.30
	ass.	0.13	2.01	0.20	3.09	0.09	0.57	0.09	0.57	0.09	0.69
	ratio	0.31	0.17	0.30	0.24	1.00	0.67	0.67	0.42	0.67	0.43
128	est.	0.04	0.20	0.04	0.37	0.08	0.27	0.05	0.17	0.05	0.17
	ass.	0.06	1.10	0.12	1.58	0.05	0.35	0.05	0.35	0.05	0.37
	ratio	0.66	0.18	0.33	0.23	1.60	0.77	1.00	0.49	1.00	0.46
	step	7	11	7	10	7	11	7	11	7	11

Table 10: Time in sec. used for assembling and error estimating for different numbers of processors, Example 3.3

proc		$\eta_{M,T}$		$\eta_{G,T}$		$\eta_{R,T;2}$		$\eta_{J,T;2}$		$\eta_{A,T;2}$	
1	est.	0.46		0.69		1.53		0.91			
	ass.	12.11		11.77		9.53		9.53			
	ratio	0.04		0.06		0.16		0.10			
4	est.	0.17		0.18		0.41		0.25		0.53	
	ass.	3.05		2.96		2.41		2.41		4.33	
	ratio	0.06		0.06		0.17		0.10		0.12	
16	est.	0.05		0.06		0.15	0.92	0.09	0.53	0.16	
	ass.	0.91		0.85		0.71	5.44	0.71	5.44	1.15	
	ratio	0.05		0.07		0.21	0.17	0.13	0.10	0.14	
64	est.	0.03	0.15	0.03	0.19	0.08	0.32	0.05	0.20	0.07	0.83
	ass.	0.33	4.46	0.39	4.83	0.23	1.70	0.23	1.70	0.44	7.10
	ratio	0.09	0.03	0.08	0.04	0.35	0.19	0.22	0.12	0.16	0.12
128	est.	0.02	0.10	0.03	0.12	0.07	0.19	0.05	0.12	0.05	0.42
	ass.	0.16	2.69	0.16	2.98	0.17	0.88	0.17	0.88	0.25	3.71
	ratio	0.13	0.04	0.19	0.04	0.41	0.22	0.29	0.14	0.20	0.11
	step	6	10	6	10	6	10	6	10	6	10

Table 11: Time in sec. used for assembling and error estimating for different numbers of processors, Example 3.4

- The presented examples illustrate that in general the h-version of adaptivity for itself cannot be the only tool for obtaining a mesh with a small number of nodes to achieve a certain error tolerance. This method has to be combined with other strategies. For example, we achieved with an a-priori oriented grid for Example 3.2 much better results than those here presented.

- The parallel overhead (communications) of the error estimators is small. It plays no role in the total parallel overhead of the solution process.

- For an efficient calculation of the error estimates there is certain amount of memory per element necessary, sometimes more than in a serial implementation.

Acknowledgement: The author wishes to acknowledge L. Tobiska for inspiring discussions on the subject of this paper and for the critical reading of the manuscript. He also wishes to thank A. Thiele for the implementation of the finite element methods in the parallel code.

References

[Bas93] P. Bastian. Parallel adaptive multigrid methods. Technical Report 93 - 60, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen der Universität Heidelberg, October 1993.

[BR93] Ivo M. Babuska and Rodolfo Rodriguez. The problem of the selection of an a posteriori error indicator based on smoothing techniques. *International Journal for Numerical Methods in Engineering*, 36(4):539 – 567, 1993.

[BW90] Randolph E. Bank and Bruno D. Welfert. A posteriori error estimators for the stokes equations: A comparison. *Comp. Meth. Appl. Mech. Engrg.*, 82:323 – 340, 1990.

[Cia78] P. Ciarlet. *The finite element method for elliptic problems.* North-Holland Publishing Company, Amsterdam - New York - Oxford, 1978.

[Mit89] William F. Mitchell. A comparision of adaptive refinement techniques for elliptic problems. *ACM Transactions on Mathematical Software*, 15(4):326 – 347, December 1989.

[Tur91] St. Turek. *Ein robustes und effizientes Mehrgitterverfahren zur Lösung der instationären, inkompressiblen 2-D Navier-Stokes-Gleichungen mit diskret divergenzfreien finiten Elementen.* PhD thesis, Universität Heidelberg, 1991.

[Ver93] R. Verfürth. A review of a posteriori error estimation and adaptive meshrefinement techniques. 1993.