A new multilevel grid deformation method

Matthias Grajewski*† Michael Köster‡ Stefan Turek §

August 7, 2008

Abstract

Recently, we introduced and mathematically analysed a new method for grid deformation [12]. This method is a generalisation of the method proposed by Liao [4, 6, 14]. In this article, we investigate the practical aspects of our method. As it requires searching the grid several times per grid point, efficient search methods are crucial for deforming grids in reasonable time, so that we propose and investigate distance and raytracing search for this purpose. By splitting up the deformation process in a sequence of easier subproblems, we significantly enhance the robustness of our deformation method. The computational speed and the accuracy of the method are improved substantially exploiting the grid hierarchy in the corresponding multilevel deformation algorithm. Being of optimal asymptotic complexity, we experience speed-ups up to a factor of 10 in our numerical tests compared to the basic deformation algorithm. This gives our new method the potential for tackling complex domains and time-dependent problems, where possibly the grid must be dynamically deformed once per time step according to the user's needs.

Keywords: mesh generation, deformation method, a posteriori error estimation, mesh adaptation

AMS classification: 65N15, 65N30, 76D05, 76D55

1 Introduction

Grid deformation, i.e. the redistribution of mesh points while preserving the topology of the mesh, has become an important alternative to element wise refinement in grid adaptation. Liao's method [4, 6] is one important member of the group of *dynamic approaches* to grid deformation. These kind of methods involve time stepping or pseudo-time stepping. Liao's method aims at deforming the given equidistributed mesh such that the spatial distribution of the cell sizes corresponds to a prescribed *monitor function*. Recently, we developed a generalisation of this method [11, 12] which permits arbitrary initial grids. Moreover, a rigorous convergence analysis is available. Both methods require the solution

*corresponding author

†Institute of Applied Mathematics, Dortmund University of Technology, Vogelpothsweg 87, D-44227 Dortmund, Germany, matthias.grajewski@mathematik.tu-dortmund.de

‡Institute of Applied Mathematics, Dortmund University of Technology, Vogelpothsweg 87, D-44227 Dortmund, Germany, michael.koester@mathematik.tu-dortmund.de

§Institute of Applied Mathematics, Dortmund University of Technology, Vogelpothsweg 87, D-44227 Dortmund, Germany, ture@featflow.de

of a single Poisson problem and of a decoupled system of initial value problems (IVPs) only. This is in contrast to many other methods for grid deformation, which necessitate expensive solution of nonlinear PDEs [5, 7, 8]. Furthermore, mesh tangling cannot occur [15] in both Liao's and our new method.

Finally, applying grid deformation in order to adapt a given computational mesh (*r-adaptivity*) offers some advantages over the widespread h-adaptivity: In many applications, local and anisotropic phenomena occur. It was shown [1, 10] that by anisotropic refinement and alignment according to such phenomena, the accuracy of the calculation can be vastly improved. Refining the such a region by subdividing existing elements only (*h-adaptivity*) may suffer from the fact that the given grid is not well-aligned. In contrast, grid deformation permits to adjust the orientation of the elements as well and thus offers additional flexibility.

It is well known that in FEM simulations grid adaptivity controlled by a posteriori error estimation is crucial for reliable and efficient computations. The widely used method of grid adaptation by allowing hanging nodes on element level, however, has severe impact on the speed of computation. Recent research [2, 18] shows that in typical adaptive FEM codes using this method of grid adaptation only a small fraction of the available processor performance of several GFlop/s can be typically used. One of the reasons for this behaviour is the extensive usage of indirect addressing in such codes which is necessary to handle the unstructured grids emerging from the adaptation procedure. On the other hand, by using local generalised tensor product meshes and thereby avoiding indirect addressing, we could achive a very significant speed-up. This has been successfully implemented in the new FEM package FEAST [2]. In this context, grid deformation is an ideal tool to grant the geometric flexibility necessary for the grid adaptation process according to a posteriori error estimators while maintaining logical tensor product structures of the grid.

In the next section, we describe in brief our basic deformation method and the corresponding convergence analysis. In section 3, the focus is placed on the aspects of implemetation, in particular, the problem of efficient grid search is adressed. Section 4 deals with enhancing the robustness of the basic grid deformation method by splitting it into several easier subproblems when necessary. In section 5, we introduce our multilevel deformation method which turns out to be superior with respect to accuracy and speed.

2 Description and numerical realisation of the basic deformation method

We first introduce some notations. A computational domain $\Omega \subset \mathbb{R}^2$ is triangulated by a conforming mesh $\mathcal{T}$ consisting of *NEL* quadrilateral elements T with size h_T. We denote the area of an element T by $m(T)$ and abbreviate the standard Lebesgue norm by $||\cdot||_0$. For a domain $D \subset \mathbb{R}^2$, the function space of k-fold continuously differentiable functions on D is referenced by $\mathcal{C}^k(D)$. For an interval I, $\mathcal{C}^{k,\alpha}(I)$, $0 < \alpha < 1$, denotes the space of functions with Hölder-continuous k-th derivatives. A domain has a $\mathcal{C}^{k,\alpha}$-smooth boundary, iff the boundary can be parameterised by a function in $\mathcal{C}^{k,\alpha}(I)$. The Jacobian matrix of a smooth mapping $\Phi : \Omega \to \Omega$ is denoted by $J\Phi$, its determinant by $|J\Phi|$.

The theoretical background of our approach – like Liao's [4, 6] – is based on Moser's work [9]. All numerical grid deformation algorithms described below aim at constructing

a bijective mapping $\Phi : \Omega \to \Omega$ satisfying

$$g(x)|J\Phi(x)| = f(\Phi(x)), \quad x \in \Omega \quad \text{and} \quad \Phi : \partial\Omega \to \partial\Omega. \tag{1}$$

The new coordinates ξ of a grid point x are computed by $\xi := \Phi(x)$. The *monitor function* f describes the distribution of the element size on the deformed mesh up to a spatially fixed constant, iff g describes the distribution of the element size on $\mathcal{T}$. Thus, g is called *area function*. Both f and g must be strictly positive on $\bar{\Omega}$. Due to $|J\Phi(x)| > 0$, mesh tangling cannot occur. Based upon [4, 6], we compute Φ in four steps. In practical

Algorithm 1: Basic grid deformation

1) Scale f or g such that

$$\int_\Omega \frac{1}{f(x)} dx = \int_\Omega \frac{1}{g(x)} dx. \tag{2}$$

For convenience, we will assume that (2) is fulfilled from now on. Let $\tilde{f}$ and $\tilde{g}$ denote the reciprocals of the scaled functions f and g.

2) Compute a grid-velocity vector field $v_h : \Omega \to \mathbb{R}^2$ by solving

$$(\nabla v_h, \nabla \varphi) = (\tilde{f} - \tilde{g}, \varphi) \quad \forall \varphi \in \mathcal{Q}_1(\mathcal{T}), \tag{3}$$

subject to homogenous Neumann boundary conditions using bilinear Lagrange elements on $\mathcal{T}$. Any other FEM space on quadrilateral or triangular meshes could be employed as well. The *recovered gradient* $G_h(v_h)$ is to approximate ∇v.

3) For each grid point x, solve the initial value problem

$$\frac{\partial \varphi(x,t)}{\partial t} = \eta_h(\varphi(x,t), t), \quad 0 \le t \le 1, \quad \varphi(x,0) = x \tag{4}$$

with

$$\eta_h(y,s) := \frac{G_h(v_h)(y)}{s\tilde{f}(y) + (1-s)\tilde{g}(y)}, \quad y \in \Omega, s \in [0,1].$$

4) Define $\Phi(x) := \varphi(x,1)$.

computations, we use the bilinear interpolant of the analytically given monitor function f instead of f itself. It has been shown [11] that this replacement does not significantly affect the deformation process. In what follows, we denote both the analytic monitor function and its interpolant by f. In a grid point x, we set $g(x)$ as the arithmetic mean of the area of the elements surrounding x. Then, we define g as the bilinear interpolant of these node values. We solve the Poisson problem (3) by conforming bilinear finite elements. The recovered gradient of its solution $G_h(v_h)$ is obtained by standard averaging techniques [21, 22].

The solution of the corresponding algebraic systems requires special care, as the solution of the Neumann problem (3) is unique up to an additive constant only. To solve (3), we use a modified multigrid method in which after every iteration the side condition $\int_\Omega v_h = 0$ is imposed by adding a suitable constant to the iteration vector.

The fully decoupled IVPs (4) are solved with standard IVP solvers. However, numerical tests [12] reveal that *any* IVP solver can be of second order at most in this situation,

0			
1/2	1/2		
1	−1	2	
	1/6	2/3	1/6

Table 1: Butcher scheme of the Kutta method of third order (RK3)

as the right hand side is just continuous and thus does not meet the smoothness requirements of high order methods. In all numerical tests presented in this article, we employ the classical third order Kutta method (RK3) which performed preferably well in previous tests. Figure 1 depicts its Butcher scheme.

In our method, the distribution of the element size is independent of the area distribution of $\mathcal{T}$. This is in contrast to Liao's methods which appears as special case for $g \equiv 1$. For the mapping Φ emerging from algorithm 1, the following holds true. We refer to [11] for details and the proof.

Theorem 1 ([12])**.** *Let the boundary of Ω be $\mathcal{C}^{3,\alpha}$-smooth and let $f, g \in \mathcal{C}^1(\Omega)$ be strictly positive in $\bar{\Omega}$. Then, if the mapping $\Phi : \Omega \to \Omega$ constructed above exists, it fulfills condition (1).*

Remark 1. *This deformation method can be applied in arbitrary dimension without any modifications. In this article, we restrict ourselves to the two-dimensional case for the sake of implementational simplicity. For investigations of the three-dimensional case, we refer to the Miemczyk's [16] and Panduranga's [17] work.*

For the grid deformation method 1, a rigorous mathematical analysis has been performed [11, 12]. This includes a proof of convergence with respect to the *quality measures*

$$Q_0 := \left|\left| \frac{f(x)}{area(x)} - 1 \right|\right|_0 \quad \text{and} \quad Q_\infty := \max_{x \in \Omega} \left| \frac{f(x)}{area(x)} - 1 \right|.$$

These measures describe the deviation of the resulting area distribution $area(\cdot)$ on the deformed grid from the prescribed area distribution f. The smaller these measures are the more accurate is the grid deformation.

When processing algorithm 1 with numerical methods, there are three error sources.

1. The deformation Poisson problem (3) is solved approximately.
2. The IVPs (4) are solved approximately.
3. The interpolation of the discontinuous cell size distribution induces a *consistency error*.

The consistency error stems from the fact that the actual cell size distribution is discontinuous and has to be interpolated in order to obtain $area(x)$. Therefore, even without any numerical error, we cannot expect one of the quality measures to be exactly zero.

For the formulation of our main convergence result, we need the following definitions.

Definition 1. *a) We call a sequence of triangulations $(\mathcal{T}_i)_{i \in I}$* edge-length regular*, iff*

$$h_i := \max_{e \in \mathcal{E}_i} |e| = \mathcal{O}(N_i^{-1/2}) \quad \forall i \in I.$$

b) The sequence of triangulations $(\mathcal{T}_i)_{i\in I}$ *is said to be* size regular, *iff*

$$\exists c, C > 0 : ch_i^2 \leq m(T) \leq Ch_i^2 \quad \forall T \in \mathcal{T}_i \, \forall i \in I. \tag{5}$$

c) An edge-length regular sequence of triangulations $(\mathcal{T}_i)_{i\in I}$ *fulfils the* similarity condition, *iff there is a function* g *with* $0 < g_{\min} \leq g(x) \leq g_{\max} < \infty$ *and* $c_s, C_s > 0$ *with*

$$\frac{1}{h_i^2} c_i m(T) = g(x) + \mathcal{O}(h_i) \quad \forall\, x \in T \quad \forall\, T \in \mathcal{T}_i \, \forall i \in I, \quad c_s \leq c_i \leq C_s. \tag{6}$$

Here, c_i *is a spatially fixed constant.*
d) A grid deformation method converges, *iff* $Q \to 0$ *for* $h_i \to 0$.

Theorem 2. *Let* $(\mathcal{T}_i)_{i\in I}$ *be a sequence of grids with grid size* $h_i \to 0$ *which fulfils condition (6) and let us denote the sequence of numerically deformed grids by* $(\tilde{\mathcal{T}}_i)_{i\in I}$. *For the monitor function* f, *we require* $0 < \varepsilon < f \in \mathcal{C}^1(\bar{\Omega})$. *Let us assume that for the approximate solution of the deformation vector field* v_h, *it holds* $||v - v_h||_\infty = \mathcal{O}(h^{1+\delta})$, $\delta > 0$. *Let* $||X_h - \tilde{X}|| = \mathcal{O}(h^{1+\delta})$ *be true for any vertex. Then,*

a) the sequence of numerically deformed meshes $(\tilde{\mathcal{T}}_i)_{i\in I}$ *is edge-length regular,*

b) $(\tilde{\mathcal{T}}_i)_{i\in I}$ *is size regular according to condition (5),*

c) the sequence of deformations converges, and $\exists c > 0$ *independent of* h, *so that*

$$Q_0 \leq ch^{\min\{1,\delta\}}, \quad Q_\infty \leq ch^{\min\{1,\delta\}}.$$

Corollary 1. *Let us assume that our numerical method for computing the IVPs (4) is of second order. Let all initial grids be size- and shape-regular and fulfil the similarity condition (6). For* f *and* $(\mathcal{T}_i)_{i\in I}$ *, the assumptions of theorem 2 may hold. Furthermore, we choose the step size* Δt *of the deformation IVPs as* $\Delta t = \mathcal{O}(h)$. *Then, if* $||v - v_h||_\infty = \mathcal{O}(h^2)$ *and if the IVP method is of second order at least,* $Q_0 \leq ch$ *and* $Q_\infty \leq ch$.

We refer to [12] for the corresponding proofs.

3 Search methods

From a theoretical point of view, the solution of the deformation IVPs (4) does not bear any difficulties and can be performed by standard methods. However, all numerical methods for IVPs require at least one evaluation of the right hand side per time step which implies the evaluation of FEM functions ($G_h(v_h)$, $\tilde{f}$ and $\tilde{g}$) in real coordinates. This requires searching the grid since one has to find the encompassing element for each evaluation point. Unfortunately, these points (except for the very first time step) may have moved to an entirely different region of the grid during the time steps already performed. Note that the evaluations of $G_h(v_h)$, $\tilde{f}$ and $\tilde{g}$ have to be carried out on the original grid, where these functions have been computed. Hence, the grid deformation algorithm 1 needs to search the whole grid (at least once) *per time step* and *per grid point*. Thus efficient searching strategies are indespensable to solve the IVPs efficiently. We propose and test three different strategies.

Brute Force We loop for each point over all elements in the grid until the element containing the point is found. For a grid consisting of N grid points the search time is

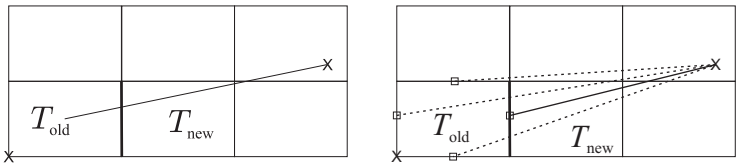

Figure 1: The principles of raytracing and distance search

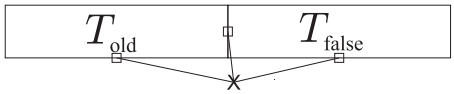

Figure 2: In this situation, distance search fails

$\mathcal{O}(N \cdot NEL) = \mathcal{O}(N^2)$, as the whole grid has to be searched in the worst case. Note that NEL and N grow with the same order.

Raytracing Being adopted from computer graphics, this method avoids searching the entire grid. We test at first if the point we search is inside the element T_{old} where it was in the previous time step. If not, we connect the center of T_{old} with the point to search for. Detecting the element edge e which intersects with this ray, we move to the element T_{new} which shares e with T_{old}. Then, setting $T_{\text{old}} := T_{\text{new}}$, we proceed as before (figure 1, left).

Special care must be taken to ensure that the ray does not intersect the element in one of its vertices as e is not unique then. In this case, we modify the starting point of the ray heuristically by disturbing the coordinates of the element center. Searching in non-convex domains requires special treatment as well. It may happen that, even though the grid point before and after the current time step is located in the domain, the ray between these points leaves the domain. In this case, we find the second intersection of this ray with the domain boundary by looping over all elements containing at least one boundary edge. We continue the search in the element containing this second intersection.

Distance search We again test at first if the point is still inside the element T_{old} where it was in the previous time step. If not, we compute the squares of the distances of the edge midpoints of T_{old} to the point we search for. The new element T_{new} is the one which is adjacent to the edge with the shortest distance (figure 1, right). However, it may happen that in certain cases the distance information leads to a wrong search direction. Then, the search may trap into an infinite loop (figure 2): Here, the comparison of the distances leads to the choice of T_{false} as next candidate. But on T_{false}, the comparison of the distances leads back of T_{old} as successor. Therefore, if during the search T_{new} is chosen as the previous element, we declare the distance search failed and start a raytracing search on T_{old} as fallback. As one step in distance search step needs less arithmetic operations than one step of raytracing search, we expect the distance search to perform faster.

The maximum length of the search path, i.e. the number of elements touched during the search process, and therefore the maximum amount for a single search process in both raytracing and distance search is $\mathcal{O}(N)$. Thus, the total search time behaves like $\mathcal{O}(N^2)$ in the worst case like for brute force search. However, almost all grids used in FEM calculations are created by regular refinement of a given coarser grid. On these grids, the

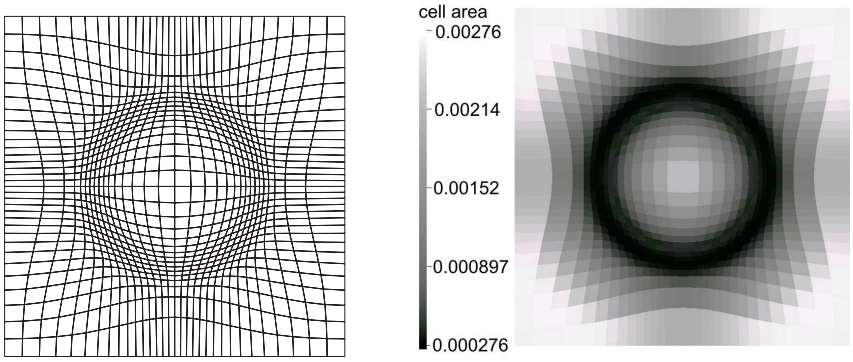

Figure 3: Resulting grid (left) and area distribution (right) for test problem 1, $\varepsilon = 0.1$, 1,024 elements

number of nodes in x- and y-direction grows with the same order leading to a maximal length of the search path of $\mathcal{O}(\sqrt{N})$. Then, the total search time grows like $\mathcal{O}(N^{3/2})$ in contrast to the brute force approach, where the search time still shows quadratic growth.

Test Problem 1. *We triangulate the unit square* $\Omega = [0,1]^2$ *with an equidistant tensor product mesh. The monitor function* f *is defined by*

$$f(x) = \min\left\{1, \max\left\{\frac{|d - 0.25|}{0.25}, \varepsilon\right\}\right\}, \quad d := \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2}. \tag{7}$$

Our choice $\varepsilon = 0.1$ *implies that on the deformed grid the largest cell has 10 times the area of the smallest one. In figure 3, we show a resulting grid consisting of 1,024 elements.*

For this test problem, we evaluate the time for searching the grid during the solution of the IVPs (4). The code is compiled using the Intel Fortran Compiler v9.1 with full optimisation and runs on an AMD Opteron 250 server equipped with a 64-bit Linux operating system. We perform 10 equidistant RK3 time steps (see section 2) as IVP solver. It requires three evaluations per component and IVP time step and thus yields 60 evaluations per grid point. We compare the search time needed for all evaluations performed in the deformation process for the three search algorithms on various levels of regular refinement. Due to the natural inaccuracy in measuring times, all tests have been repeated until the relative error of the mean value of the measured times fell below 1%.

The search times are displayed in figure 4 (left). Tests skipped due to overly long computational times are marked with "-". Both distance and raytracing search are by far faster than the brute force approach, which by its quadratic search time (indicated by the slope of 2) leads to inappropriate search times ($\approx$ 3 h in our example). In contrast, the search times of distance and raytracing search grow even smaller than $\mathcal{O}(N^{3/2})$. However, the slope obviously increases to 1.5 with grid refinement. On very fine grids, where the search paths are longer than on coarse ones, distance search is superior: For $NEL \approx 10^6$, distance search is roughly 30% faster than the raytracing approach.

Potentially, hierarchical search methods based on spatial partitions feature even higher speed, as the search time per evaluation needs only $\mathcal{O}(\log N)$ operations yielding a total search time of $\mathcal{O}(N \log N)$. However, these methods require sufficiently long

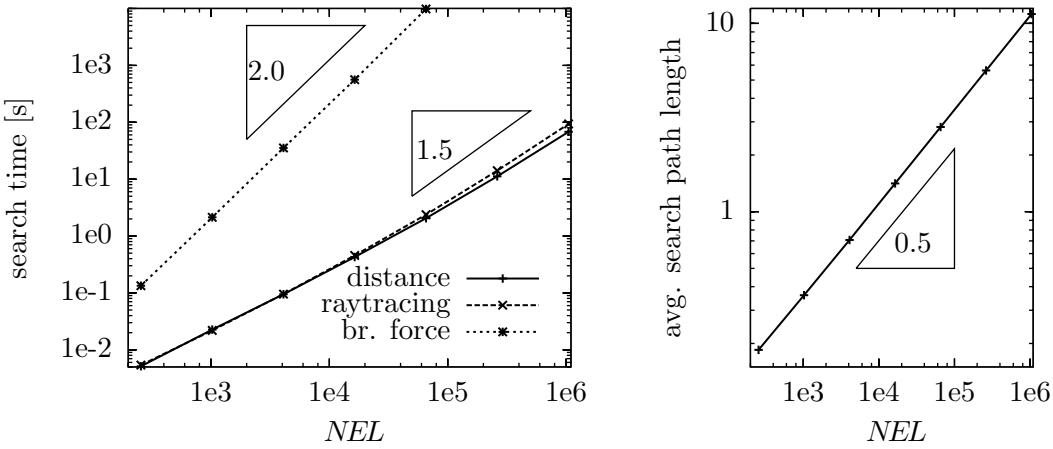

Figure 4: Search times (left) and average search path lengths (right) vs. number of elements *NEL* for test problem 1, 10 time steps

search paths to perform superior because of their inherent overhead of constructing the search hierarchies. Here, the length of the search path is defined by the number of element changes during one search. Consequently, the search path has length zero, if the search ends in the same element where it has started. For test problem 1, we present in figure 4 (right) the average length of the search paths for searching the grid points during deformation. We ignore the searches of the intermediate points occurring inside RK3 as well as searches of boundary points. The average search path length doubles in every refinement step as expected. That arises from the fact that a grid point on a fine grid is moved (almost) to the same coordinates than on a coarser grid, but the element size is halved in each refinement. Even on very fine grids, the search path length is rather small. This shows that, at least for the example presented here, *the search paths are too small to expect significant speed-ups using hierarchical searching even on very fine grids.*

4 Enhancing robustness

We now investigate the practical behaviour of the basic grid deformation method 1 and put the emphasis on robustness aspects.

The grid resulting from test problem 1 (figure 3) demonstrates that this test problem can be easily solved by algorithm 1. However, in the case of monitor functions with very steep gradients or monitor functions implying extreme variations in element size, it may happen that the corresponding vector field v cannot be resolved properly on the undeformed given grid. In this case, non-convex elements can appear. This is not due to theoretical limitations of the method, but due to the numerical error induced by the approximate solution of the Poisson problem (3). In this context, the error produced by the numerical solution of the IVP (4) seems to be far less critical. Employing test problem 1, we now investigate the robustness of our deformation method with respect to desired sharp concentrations of grid cells in one region of the mesh. The computations are carried out on tensor product meshes of various levels of refinement. As IVP solver, we choose

NEL	INT	SPR	PPR
4,096	$1.9 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$
16,384	$2.0 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$
65,536	$1.5 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$
262,144	$1.2 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$
1,048,576	$9.0 \cdot 10^{-3}$	$9.0 \cdot 10^{-3}$	$9.0 \cdot 10^{-3}$

Table 2: Minimal shape parameters $\varepsilon_{\min}$ for which test problem 1 can be computed

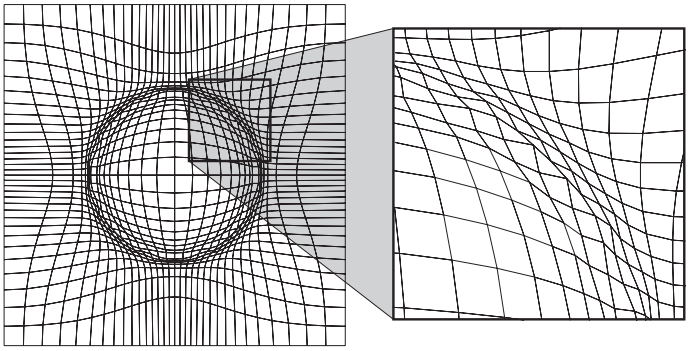

Figure 5: Resulting grid for test problem 1, $\varepsilon = 0.018$, 1,024 elements.

the RK3 method with a constant time step size of 0.02. Decreasing the shape parameter ε in the monitor function (7) leads to an increasing concentration of the mesh around the circle. In table 2, we show the lowest shape parameter $\varepsilon_{\min}$ for which the deformation process leads to valid grids with respect to the grid size as well as to the method for gradient recovery. We compare standard averaging (INT), the superconvergent path recovery technique by Zienkiewicz and Zhu (SPR) [22] and the polynomial preserving patch recovery technique (PPR) by Zhang [21]. All these techniques lead to recovered gradients of second order accuracy on regular meshes. To determine $\varepsilon_{\min}$, we compute test problem 1 repeatedly and decrease ε by 0.001 in every run. The results exhibit that regardless of the level of refinement, the basic algorithm 1 fails if ε falls below ≈ 0.01. For coarser grids with 1024 elements or even less, the monitor function cannot be properly resolved on the initial mesh which disturbs the measurements. Therefore we ignore this particular case. The minimal shape parameters in table 2 do not show any strong dependency on the grid level although on increasing high levels, $\varepsilon_{\min}$ seems to decrease. It is almost independent of the choice of the recovery method for this test problem. For $\varepsilon = 1.8 \cdot 10^{-2}$ and $NEL = 1{,}024$, the elements around the circle are nearly non-convex, the maximal angle measured in this grid is 179.89° (figure 5).

Repeating this numerical test with a smaller constant step size of 0.002 yields exactly the same results. Because of this, the main numerical error in the deformation method must be introduced by the approximate solution of the deformation PDE (3).

Enhancing the robustness by solving the deformation Poisson problem (3) more accurately, e.g. by using a highly refined mesh would require overly demanding calculations. Instead, we split the deformation up into several less demanding *adaptation steps* accord-

ing to a *blended monitor function* f_s defined by

$$f_s(x) := sf(x) + (1-s)g(x), \quad s \in [0,1] \tag{8}$$

instead of f itself. We now iterate algorithm 1 increasing s in every step such that the composition of all adaptation steps yields the desired deformed grid. For compatibility reasons, we require $\int_\Omega f = \int_\Omega g$ here.

Algorithm 2: RobustDeformation

Input: f, g, γ_0, *GRID*
Output: *GRID*

$n_a :=$ `CompNa(`f, g, γ_0`)`;
$S :=$ `CompBlendpars(`f, g, γ_0`)`;
for $i = 1$ **to** n_a **do**
  GRID := `BasicDeformation(`$f_{S(i)}$, *GRID*`)`
end

Here, the question arises how to choose the *blending parameter* s in formula (8) and how many adaptation steps n_a to perform. In algorithm 2, this corresponds to defining the functions `CompNa` and `CompBlendPars` which determines for all adaptation steps the blending parameter $S(i), 1 \leq i \leq n_a$. The user-defined parameter γ_0 determines the maximal amount of deformation per adaptation step, we elaborate on this below. For the monitor function f defined in equation (7), it turns out that ε, which describes the ratio of the area of the smallest and largest elements on the deformed grid is a certain measure for the difficulty of the deformation: For $\varepsilon = 1$, the deformation is trivial as $\Phi = \mathrm{Id}$, the choice $\varepsilon = 0.1$ served as first test example; for $\varepsilon = 0.01$, the basic deformation method fails as demonstrated above. Deforming a grid according to f with $\varepsilon = 0$ would lead to elements with area zero and is thus impossible at all. Let now $f_{\min}$ and $f_{\max}$ denote the maximum and the minimum of an arbitrary monitor function f in Ω. With $\varepsilon \approx f_{\min}/f_{\max}$ for the monitor function (7), we interpret the ratio $f_{\max}/f_{\min}$ as indicator for the numerical difficulty of the deformation problem. These considerations are valid when starting from an initial mesh with constant area distribution only. We however expect algorithm 1 to work properly when tackling the test problem 1 with small ε on a suitably pre-deformed grid, where the actual deformation is almost trivial. Here, it holds $f(x) \approx g(x)$ regardless of the properties of both f and g. Thus, we consider the deviation of $f(x)/g(x)$ from a constant value as indicator of the "amount of numerical difficulties" of the deformation problem. Rearranging the transformation equation (1) to $|J\Phi(x)| = f(\Phi(x))/g(x)$ suggests to define

$$\gamma' := \frac{\max_{x\in\Omega} \frac{f(\Phi(x))}{g(x)}}{\min_{x\in\Omega} \frac{f(\Phi(x))}{g(x)}} \geq 1$$

as heuristical difficulty measure. Unfortunately, γ' is unsuitable for practical computations as it relies on the unknown transformation Φ. Therefore, we neglect the influence of Φ and define

$$\gamma := \frac{(f(x)/g(x))_{\max}}{(f(x)/g(x))_{\min}} \geq 1$$

as indicator for the numerical problems to expect during deformation. For the trivial deformation problem $f = g$, it holds $\gamma = \gamma' = 1$. In the special case of test problem 1, we have $g = 1$ after scaling. Hence, it then holds $\gamma = f_{\max}/f_{\min} \approx \varepsilon^{-1}$, and the initial "difficulty measure" is recovered.

We use the blending in formula (8) in order to *reduce γ such that every adaptation step according to f_s in algorithm 2 can be treated by the basic algorithm 1.* Let us assume that algorithm 1 is able to produce valid grids for all settings with $1 < \gamma < \gamma_0$ and take for granted that for the given monitor function $\bar{f}$ and the given current area distribution $\bar{g}$, it holds $\bar{\gamma} > \gamma_0$. In this situation, n_a is set to

$$n_a = \left\lceil \frac{\ln(\bar{\gamma})}{\ln(\gamma_0)} \right\rceil, \tag{9}$$

$\lceil x \rceil := \min_{k \in \mathbb{Z}} \{k \geq x\}$ denoting the ceiling function. The blending parameter $S(i)$ (compare formula (8)) in the i-th grid adaptation step is chosen such that

$$\gamma_i = \left(\sqrt[n_a]{\bar{\gamma}} \right)^i \leq (\gamma_0)^i \tag{10}$$

holds. Here, γ_i is computed a priori before any deformation has been performed. Starting on the undeformed mesh, we perform a sequence of n_a deformation steps where each of these steps aims to satisfy

$$g_i(x)|\Phi_i(x)| = f_{S(i)}(\Phi_i(x)), \quad i = 1, \ldots n_a. \tag{11}$$

Because of

$$g_i(x) = f_{s_{i-1}}(\Phi_{i-1}(x)), \tag{12}$$

we conclude that for the composition transformation $\Phi := \Phi_{n_a} \circ \ldots \circ \Phi_1$ which implicitly has been applied to the initial grid, it holds

$$\begin{aligned}
|J\Phi(x)| &= |J(\Phi_{n_a} \circ \ldots \circ \Phi_1(x))| \\
&= |J(\Phi_{n_a})(\Phi_{n_a-1} \circ \ldots \circ \Phi_1(x)) \cdot J(\Phi_{n_a-1} \circ \ldots \circ \Phi_1(x))| \\
&= |J(\Phi_{n_a})(\Phi_{n_a-1} \circ \ldots \circ \Phi_1(x))| \ldots |J\Phi_1(x)| \\
&\overset{(11)}{=} \frac{f_{s_{n_a}}(\Phi_{n_a} \circ \ldots \circ \Phi_1(x))}{g(\Phi_{n_a-1} \circ \ldots \Phi_1(x))} \frac{f_{s_{n_a-1}}(\Phi_{n_a-1} \circ \ldots \circ \Phi_1(x))}{g(\Phi_{n_a-2} \circ \ldots \Phi_1(x))} \cdots \frac{f_{s_1}(\Phi_1(x))}{g(x)}.
\end{aligned}$$

This leads, if $s_{n_a} = 1$, to the deformation equation (1). By virtue of equations (10), (11) and (12), every single deformation step is computable with algorithm 1.

Lemma 1. *The blending parameter $S(i)$ in the i-th adaptation step has to be chosen as*

$$S(i) = \frac{\left(\sqrt[n_a]{\bar{\gamma}} \right)^i - 1}{\left(\frac{f}{g} - 1 \right)_{\max} - \gamma_i \left(\frac{f}{g} - 1 \right)_{\min}}. \tag{13}$$

In the case of an equidistributed initial grid, the computation of $S(i)$ simplifies to

$$S(i) = \frac{\left(\sqrt[n_a]{\bar{\gamma}} \right)^i - 1}{f_{\max} + \left(\sqrt[n_a]{\bar{\gamma}} \right)^i (1 - f_{\min}) - 1}. \tag{14}$$

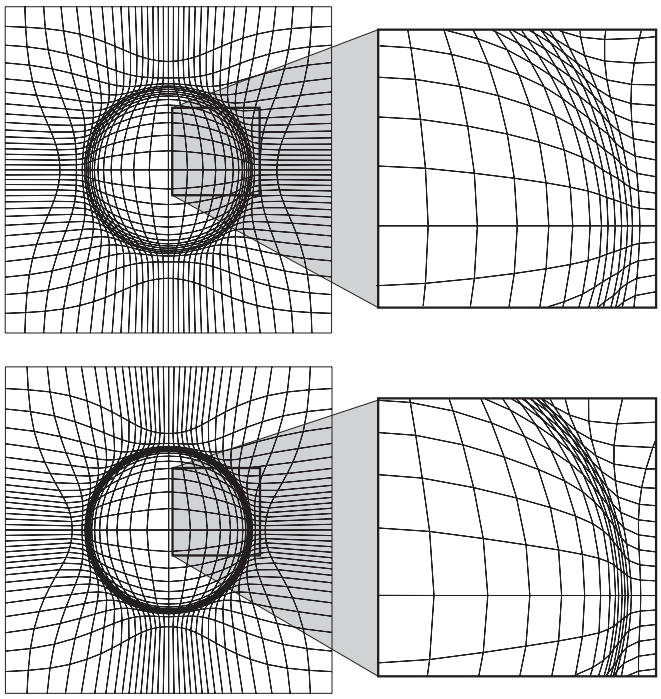

Figure 6: Resulting grid for test problem 1, $\varepsilon = 0.018$ (top) and, $\varepsilon = 0.005$ (bottom) 1,024 elements. The comparison with figure 5 demonstrates the improved grid quality.

Proof. We obtain (13) by a straightforward calculation starting from

$$\frac{(f_{S(i)}/g)_{\max}}{(f_{S(i)}/g)_{\min}} \stackrel{!}{=} \gamma_i.$$

Statement (14) follows directly from equation (13). □

With the improved algorithm 2, it is possible to compute test problem 1 for $\varepsilon = 0.018$ and even $\varepsilon = 0.005$ (figure 6) without any difficulties. We prescribe $\gamma_i = 10$ resulting in 2 adaptation steps and compute the test problem on a tensor product grid with 1,024 elements. In contrast to this, algorithm 1 leads to invalid grids in this case regardless of the IVP solver, the number of time steps and the method for gradient recovery.

The choice of γ_0 is of largely heuristic nature. The numerical tests presented lead to the practical guess of $\gamma_0 = 10$ which is chosen for all subsequent tests. For practical computations, the break-up into several less harsh deformation problems guided by γ and the choice of adaptation steps described above leads to a significant gain of robustness and thus enlarges the class of solvable deformation problems considerably.

Remark 2. *Computing test problem 1 with* $\varepsilon = 0.001$ *still leads to tangled grids in spite of the improvements of the deformation algorithm. This is due to the fact that our grid deformation method is able to control element size, but not element shape. With decreasing* ε*, the minimal angle in the resulting grid tends to 0 and the maximal angle to* π*. Then, even very small numerical errors during deformation may lead to non-convex elements. As a remedy, one can apply grid smoothing after every adaptation step. Even*

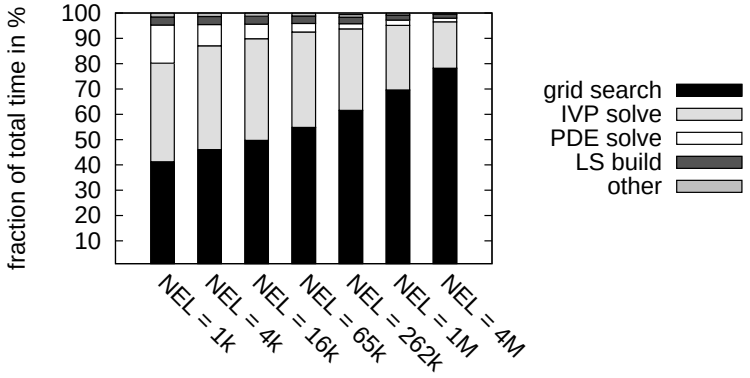

Figure 7: Relative timings for selected components of grid deformation method 2

though numerical tests provide encouraging results, the application of grid smoothing has heuristical character and depends very much on the given deformation problem.

5 Multilevel deformation

For practical computations, besides accuracy and robustness, the runtime behaviour of our deformation algorithm is important, as the deformation time may only be a small fraction of the overall runtime. We now test every main component of algorithm 2 with respect to numerical complexity and time consumption.

To solve the global Poisson problem (3) in the deformation method, we employ multigrid solvers which grow linearly with the number of unknowns and therefore are of optimal complexity, at least for reasonable meshes. The same applies obviously for the actual IVP solve, given a constant number of IVP time steps. In contrast to this, the amount for searching the grid during the IVP solve behaves like $N^{3/2}$ (section 3). Thus the time for searching the grid will dominate the overall deformation time on sufficiently fine grids.

Remark 3. *In order to obtain convergence, it is necessary to double the number of time steps per refinement step of the initial grid (see section 2, corollary 1). Because of this, the search path length remains bounded and the asymptotic complexity is $\mathcal{O}(N^{3/2})$ like in the case of constant time step size. However, the former choice of the time step size implies by far longer computational times than the latter one.*

We now aim at accelerating the deformation method in the case of constant time step size and neglect in a first step the convergence behaviour. We compute test problem 1 with $\varepsilon = 0.1$ on an equidistant tensor product grid with deformation algorithm 2 and perform one adaptation step according to formula (9). No postprocessing is employed. As IVP solver, we apply 10 equidistant RK3 steps and search the grid by raytracing search. We solve the deformation PDE (3) with multigrid which performs four line ADI steps [19] for pre- and postsmoothing, respectively. The corresponding coarse grid problems are solved with the method of conjugate gradients (CG). The multigrid iteration stops when the norm of the residuum falls below 10^{-9}. For the tests, we ultilise the same computer and infrastructure as before. The relative timings (figure 7) confirm the anticipated dominance of the time for raytracing search caused by its superlinear growth. Searching the grid ("grid search") takes approx. 60% of the overall computational time on the finest

grid, whereas the actual IVP solve ("IVP solve") and the assembly of the linear system for the PDE (3) ("LS build") form a small part only. The same holds for its solve time ("PDE solve"). All other operations in grid deformation like scaling the monitor function, computing the current area distribution or computing quality measures are embraced in "other".

Accelerating the deformation method therefore implies decreasing the overall search time. From now on, we assume and exploit that our sequence of initial grids is constructed by repeated regular refinement of one coarse grid. This is a much stronger condition than the similarity condition (6) considered so far. We assume that at least on fine meshes, regular refinement and grid deformation almost commute, i.e. that the grids $\mathcal{T}_1$ resulting from deformation after one step of regular refinement and $\mathcal{T}_2$ coming from one step of regular refinement after deformation are very similar. Note that compared to the deformation algorithms presented so far, the average search path length is cut in half for $\mathcal{T}_2$. Then, applying grid deformation to the pre-deformed grid $\mathcal{T}_2$ will not lead to significant changes of the positions of the vertices. Therefore we can expect short average search paths for this deformation step resulting in short search times on the finest grid. We define the search path length as in section 3. Iterating this two-level approach leads to the *multilevel deformation algorithm* 3.

Algorithm 3: multilevel deformation

Input: f, g, γ_0, $i_{\min}$, $i_{\max}$, i_{incr}, $N_{\text{pre}}(i)$, $GRID$
Output: $GRID$

$GRID_{i_{\min}}$:= `Restrict(`$GRID$, $i_{\min}$`)`;
for $i = i_{\min}$ **to** $i_{\max}, i_{\text{incr}}$ **do**
    $GRID_i$:= `PreSmooth(`$GRID_i, N_{\text{pre}}(i)$`)`;
    $GRID_i$:= `RobustDeformation(`$f, g, \gamma_0, GRID_i$`)`;
    if $i < i_{\max}$ **then** $GRID_{i+1}$:= `Prolongate(`$GRID_i$`)`
end
$GRID := GRID_{i_{\max}}$;

By `Prolongate` and `Restrict` we denote regular refinement and coarsening of our computational grid, respectively. The coarsening is performed by omitting all vertices which do not belong to the coarse grid. By `PreSmooth`, we denote a generic grid smoothing strategy, e.g. Laplacian smoothing. `Prolongate`, `Restrict` and `PreSmooth` must not be confused with the similarly named building blocks of a multigrid solver for linear systems.

In what follows, we refer to the deformation algorithm 2 as "one-level deformation" in contrast to the "multilevel deformation" (algorithm 3). We again consider test problem 1 with the same settings as above, but using algorithm 3 instead. We set $i_{\text{incr}} = 1$ and $i_{\min} = 3$ leading to an initial grid consisting of 256 elements. The parameter $N_a(i_{\min})$ is chosen adaptively by formula (9), $N_a(i) = 1 \, \forall i > i_{\min}$. We set $N_{\text{pre}} \equiv 2$. As desired, the average search path length remains bounded in multilevel deformation (figure 8, right) even for a step size constant with respect to the level of refinement. Based upon this numerical result, we now prove *optimal complexity* of the multilevel deformation.

Lemma 2. *We assume that in multilevel deformation, the average search path length is bounded independently of the level of refinement. Let N denote the number of nodes in*

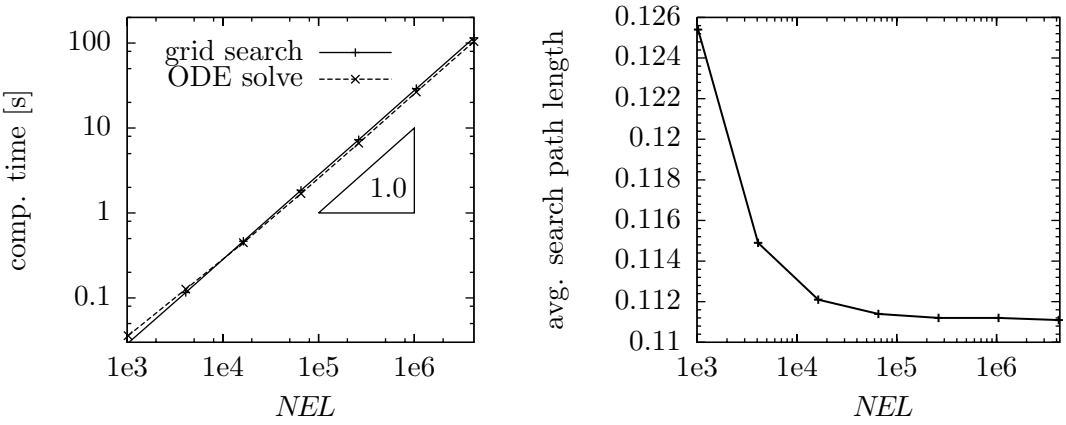

Figure 8: Computational times for grid search and IVP solve for multilevel deformation (left) and average search path length (right) vs. number of elements *NEL*

the grid. Then, the total number of IVP time steps N_{solve} as well as the total number N_{search} of point-in-element-tests during search grow at most like N.

Proof. Let us denote the number of nodes on the grid on level $i_{\min}$ by N_0. Further, we assume $i_{\text{incr}} = 1$ without loss of generality. Then, the grid on the finest level consists of $N_0 \cdot 4^{(i_{\max}-i_{\min})}$ elements. As all grids on the levels in between $i_{\min}$ and $i_{\max}$ are deformed, totally $\sum_{i=i_{\min}}^{i_{\max}} N_0\, 4^{i-i_{\min}}$ grid points are moved, and thus

$$\begin{aligned} N_{solve} &\leq C \sum_{i=i_{\min}}^{i_{\max}} N_0 4^{i-i_{\min}} \leq C N_0 N \sum_{i=i_{\min}}^{i_{\max}} \frac{1}{N_0 4^{i_{\max}-i_{\min}}} 4^{i-i_{\min}} \\ &= CN \sum_{i=i_{\min}}^{i_{\max}} \left(\frac{1}{4}\right)^{i-i_{\max}} = CN \sum_{k=0}^{i_{\max}-i_{\min}} \left(\frac{1}{4}\right)^{k} \leq \frac{4}{3} CN \end{aligned}$$

where C stands for the maximum over the number of time steps. This proves the first part of the statement. The second one follows from the boundedness of the search path length. □

For test problem 1, the computational time for grid search and IVP solve grows linearly (cf. figure 8) when applying multilevel deformation in contrast to $\mathcal{O}(N^{3/2})$ for the one-level deformation. Regarding the computational time as measure for their numerical complexity, these findings confirm the statement of lemma 2.

The multilevel deformation performs much faster than the one-level deformation method on fine grids (figure 9). Moreover, it produces more accurate results as indicated by smaller quality numbers (table 5). This confirms our assumption, that on fine grids, regular refinement and grid deformation almost commute. For coarse grids up to 65,000 elements, the quality measures of the multilevel deformation are inferior to the ones obtained from one-level deformation. This likely stems from the poor resolution of the desired area distribution on coarse grids such that the deformation step on the fine level can only partially be regarded as correction step. However, multilevel deformation is intended for very fine grids only.

NEL	Q_0	Q_∞	$h_{\min}$	$\alpha_{\min}$	$\alpha_{\max}$	runtime [s]
1,024	$5.817 \cdot 10^{-2}$	$2.630 \cdot 10^{-1}$	$1.10 \cdot 10^{-2}$	41.13	140.41	$1.02 \cdot 10^{-1}$
16,384	$7.833 \cdot 10^{-3}$	$6.622 \cdot 10^{-2}$	$2.56 \cdot 10^{-3}$	36.80	143.62	$8.46 \cdot 10^{-1}$
262,144	$1.422 \cdot 10^{-3}$	$1.668 \cdot 10^{-2}$	$6.32 \cdot 10^{-4}$	35.81	144.32	$1.07 \cdot 10^{1}$
1,048,576	$6.626 \cdot 10^{-4}$	$8.417 \cdot 10^{-3}$	$3.15 \cdot 10^{-4}$	35.68	144.37	$3.97 \cdot 10^{1}$
4,194,304	$3.190 \cdot 10^{-4}$	$4.237 \cdot 10^{-3}$	$1.57 \cdot 10^{-4}$	35.63	144.41	$1.60 \cdot 10^{2}$
1,024	$6.449 \cdot 10^{-2}$	$3.004 \cdot 10^{-1}$	$1.13 \cdot 10^{-2}$	44.66	136.92	$9.09 \cdot 10^{-2}$
16,384	$8.380 \cdot 10^{-3}$	$7.295 \cdot 10^{-2}$	$2.48 \cdot 10^{-3}$	38.21	142.21	$8.26 \cdot 10^{-1}$
262,144	$1.476 \cdot 10^{-3}$	$1.836 \cdot 10^{-2}$	$6.10 \cdot 10^{-4}$	37.20	142.98	$1.06 \cdot 10^{1}$
1,048,576	$6.808 \cdot 10^{-4}$	$9.191 \cdot 10^{-3}$	$3.05 \cdot 10^{-4}$	37.10	142.99	$4.12 \cdot 10^{1}$
4,194,304	$3.266 \cdot 10^{-4}$	$4.707 \cdot 10^{-3}$	$1.52 \cdot 10^{-4}$	37.05	142.98	$1.62 \cdot 10^{2}$
1,024	$8.108 \cdot 10^{-2}$	$3.045 \cdot 10^{-1}$	$1.23 \cdot 10^{-2}$	46.33	135.65	$5.97 \cdot 10^{-2}$
16,384	$9.091 \cdot 10^{-3}$	$8.277 \cdot 10^{-2}$	$2.53 \cdot 10^{-3}$	37.84	142.52	$8.15 \cdot 10^{-1}$
262,144	$1.535 \cdot 10^{-3}$	$2.000 \cdot 10^{-2}$	$6.22 \cdot 10^{-4}$	36.71	143.46	$1.10 \cdot 10^{1}$
1,048,576	$7.050 \cdot 10^{-4}$	$9.987 \cdot 10^{-3}$	$3.10 \cdot 10^{-4}$	36.61	143.47	$4.16 \cdot 10^{1}$
4,194,304	$3.373 \cdot 10^{-4}$	$5.170 \cdot 10^{-3}$	$1.55 \cdot 10^{-4}$	36.56	143.48	$1.62 \cdot 10^{2}$
NEL	Q_0	Q_∞	$h_{\min}$	$\alpha_{\min}$	$\alpha_{\max}$	runtime [s]
1,024	$1.380 \cdot 10^{-1}$	$6.899 \cdot 10^{-1}$	$3.67 \cdot 10^{-3}$	15.77	168.34	$1.47 \cdot 10^{-1}$
16,384	$2.047 \cdot 10^{-2}$	$1.557 \cdot 10^{-1}$	$4.95 \cdot 10^{-4}$	8.39	172.52	$1.10 \cdot 10^{0}$
262,144	$3.117 \cdot 10^{-3}$	$3.735 \cdot 10^{-2}$	$1.19 \cdot 10^{-4}$	7.92	172.31	$1.35 \cdot 10^{1}$
1,048,576	$1.370 \cdot 10^{-3}$	$1.872 \cdot 10^{-2}$	$5.94 \cdot 10^{-5}$	7.93	172.18	$4.88 \cdot 10^{1}$
4,194,304	$6.371 \cdot 10^{-4}$	$9.348 \cdot 10^{-3}$	$2.96 \cdot 10^{-5}$	7.92	172.13	$2.01 \cdot 10^{2}$

Table 3: Quality measures Q_0, Q_∞, minimal element size $h_{\min}$ as well as minimal and maximal angles $\alpha_{\min}$, $\alpha_{\max}$ and runtime (AMD Opteron 250), $i_{\min} = 2$ (upper part), $i_{\min} = 3$ (medium part) and $i_{\min} = 4$ (lower part) for selected levels of refinement, test problem 1 with $\varepsilon = 0.1$ and $i_{min} = 4, \varepsilon = 0.01$ (lower table)

The multilevel deformation requires the user to choose the new parameters $i_{\min}$, i_{incr} and $N_{\text{pre}}(i)$. We now investigate numerically the dependence of the accuracy of the deformation process and the quality of the resulting grid from the choice of $i_{\min}$ and N_{pre}. We again compute test problem 1 with multilevel deformation algorithm 3 with the same settings as above for $i_{\min} = 2, \ldots 4$ yielding initial grids consisting of 64, 256 and 1,024 elements, respectively. The influence of $i_{\min}$ is small and decreases with growing grid level (table 3). This holds true both for the quality measures and the quantities measuring the grid quality. The difference of the runtimes turned out to be insignificant. These experiments indicate that the multilevel algorithm 3 is robust with respect to the choice of $i_{\min}$ given that the deformation problem is solvable on the initial grid.

We set $i_{\min}$ to 3, but now vary the number of Laplacian presmoothing steps $N_{\text{pre}}(i)$. We compute test problem 1 with the same parameter settings as above and choose the number of Laplacian presmoothing steps uniformly for all grid refinement levels to 2 and 4. As the computational time for grid smoothing contributes only insignificantly to the overall runtime, we omit the runtime for these tests. The multilevel deformation algorithm turns out to be robust with respect to the choice of presmoothing steps as well (table 4). Increasing the number of smoothing steps from 2 to 4 leads to a significant

NEL	Q_0	Q_∞	$h_{\min}$	$\alpha_{\min}$	$\alpha_{\max}$
1,024	$6.237 \cdot 10^{-2}$	$2.562 \cdot 10^{-1}$	$1.13 \cdot 10^{-2}$	44.66	136.92
16,384	$8.329 \cdot 10^{-3}$	$6.310 \cdot 10^{-2}$	$2.48 \cdot 10^{-3}$	38.21	142.21
262,144	$1.475 \cdot 10^{-3}$	$1.594 \cdot 10^{-2}$	$6.10 \cdot 10^{-4}$	37.20	142.98
1,048,576	$6.807 \cdot 10^{-4}$	$8.042 \cdot 10^{-3}$	$3.05 \cdot 10^{-4}$	37.10	142.99
4,194,304	$3.266 \cdot 10^{-4}$	$4.323 \cdot 10^{-3}$	$1.52 \cdot 10^{-4}$	37.05	142.98
1,024	$5.887 \cdot 10^{-2}$	$2.427 \cdot 10^{-1}$	$1.12 \cdot 10^{-2}$	44.24	137.43
16,384	$6.859 \cdot 10^{-3}$	$6.082 \cdot 10^{-2}$	$2.50 \cdot 10^{-3}$	38.29	142.20
262,144	$8.207 \cdot 10^{-4}$	$1.518 \cdot 10^{-2}$	$6.14 \cdot 10^{-4}$	37.44	142.72
1,048,576	$2.876 \cdot 10^{-4}$	$7.651 \cdot 10^{-3}$	$3.06 \cdot 10^{-4}$	37.32	142.76
4,194,304	$1.036 \cdot 10^{-4}$	$3.892 \cdot 10^{-3}$	$1.53 \cdot 10^{-4}$	37.27	142.77

Table 4: Quality measures Q_0, Q_∞, minimal element size $h_{\min}$ as well as minimal and maximal angles $\alpha_{\min}$ and $\alpha_{\max}$, $N_{\text{pre}} = 2$ (upper part) and $N_{\text{pre}} = 4$ (lower part) for selected levels of refinement, test problem 1 with $\varepsilon = 0.1$

	Q_0			Q_∞		
NEL	one-lev	$i_{\text{incr}} = 1$	$i_{\text{incr}} = 2$	one-lev	$i_{\text{incr}} = 1$	$i_{\text{incr}} = 2$
1,024	$8.11 \cdot 10^{-2}$	$6.24 \cdot 10^{-2}$	$7.96 \cdot 10^{-2}$	$3.05 \cdot 10^{-1}$	$2.56 \cdot 10^{-1}$	$2.90 \cdot 10^{-1}$
4,096	$3.08 \cdot 10^{-2}$	$2.23 \cdot 10^{-2}$	$2.36 \cdot 10^{-2}$	$1.75 \cdot 10^{-1}$	$1.25 \cdot 10^{-1}$	$1.39 \cdot 10^{-1}$
16,384	$1.08 \cdot 10^{-2}$	$8.33 \cdot 10^{-3}$	$1.10 \cdot 10^{-2}$	$8.26 \cdot 10^{-2}$	$6.31 \cdot 10^{-2}$	$8.28 \cdot 10^{-2}$
65,536	$3.74 \cdot 10^{-3}$	$3.39 \cdot 10^{-3}$	$3.37 \cdot 10^{-3}$	$4.67 \cdot 10^{-2}$	$3.19 \cdot 10^{-2}$	$3.53 \cdot 10^{-2}$
262,144	$3.61 \cdot 10^{-3}$	$1.48 \cdot 10^{-3}$	$1.80 \cdot 10^{-3}$	$3.69 \cdot 10^{-2}$	$1.59 \cdot 10^{-2}$	$2.07 \cdot 10^{-2}$
1,048,576	$5.71 \cdot 10^{-3}$	$6.81 \cdot 10^{-4}$	$7.03 \cdot 10^{-4}$	$5.18 \cdot 10^{-2}$	$8.04 \cdot 10^{-3}$	$8.61 \cdot 10^{-3}$
4,194,304	$7.02 \cdot 10^{-3}$	$3.28 \cdot 10^{-4}$	$3.99 \cdot 10^{-4}$	$6.27 \cdot 10^{-2}$	$4.32 \cdot 10^{-3}$	$5.57 \cdot 10^{-3}$

Table 5: Quality measures for test problem 1 computed with one-level deformation (left part) and multilevel deformation 3 with $i_{\text{incr}} = 1$ (medium part) and $i_{\text{incr}} = 2$ (right part)

improvement of Q_0 which is particularly notable on very fine levels. However, Q_∞ changes much less, and the difference in the other grid-related quantities displayed are minor to insignificant. It is possible to set presmoothing aside at all, but this lowers the grid quality and deformation accuracy. Overall, we conclude that the multilevel deformation benefits from presmoothing, but is robust with respect to the number of presmoothing steps, i.e. changing their number will not lead to significantly different resulting grids.

As obviously the deformation steps on the finer grids change the mesh only minimally, it seems favorable to omit some "intermediate" levels for further acceleration. To do so, we enlarge set $i_{\min} = 2$ in algorithm 3 and repeat the tests from above. For even levels of refinement, the starting level $i_{\min}$ is set to 4, for odd ones to 3 resulting in initial grids with 256 and 1024 elements, respectively. This is to guarantee that the actual level increment is always two. The corresponding quality measures (table 5) show a slight decline compared to the case of $i_{\text{incr}} = 1$, but they are still much better than the ones of the one-level deformation on grids with 10^6 elements and more (cf. table 5).

The comparison of the overall runtimes for the different deformation algorithms in figure 9 demonstrates that by multilevel deformation, a significant speed-up can be achieved. Here, we denote by OneLevConst the variant of the one-level deformation with constant

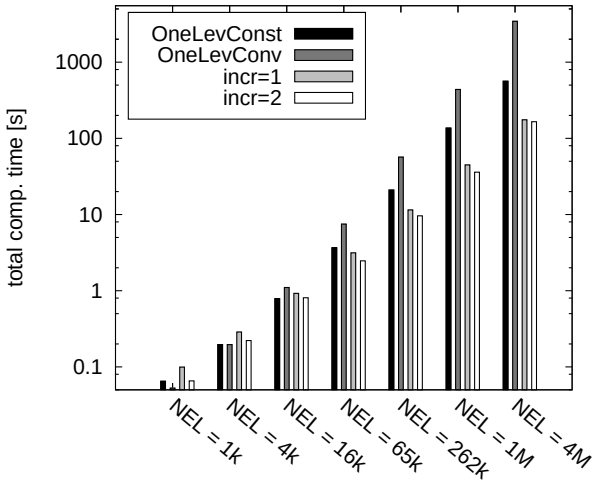

Figure 9: Runtime comparisons for one-level deformation with constant and increasing step size and multilevel deformation with level increment 1 and 2

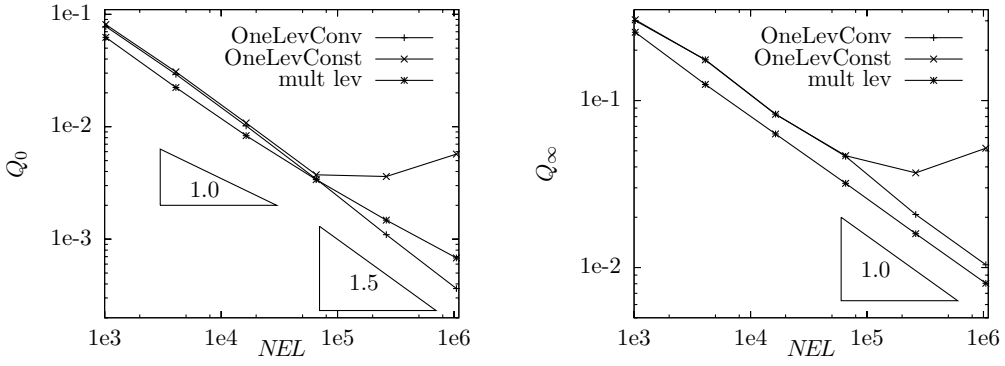

Figure 10: Quality measures vs. *NEL* for one-level and multilevel deformation methods

size of IVP time steps independent of the refinement level. OneLevConv stands for the variant where the number of time steps doubles per refinement in order to achieve convergence. The runtimes are measured on the same computer and with the same settings as above. The tests were repeated until the standard deviations dropped below 1%. On high levels of refinement, OneLevConv is up to 3 times slower than OneLevConst confirming the statements of remark 3. The speed-up of the multilevel deformation increases with the level of refinement due to the different orders of complexity.

Our numerical experiments revealed the superior accuracy of the multilevel deformation regardless of the setting of its free parameters. We now revisit the convergence aspects of section 2 and compare the results of this section with the ones from the different variants of the one-level methods considered so far. These are OneLevConv whose observed convergence relies on doubling the time steps per regular refinement and OneLevConst with a fixed time step size. As the IVP-induced error does not decrease, OneLevConst does not converge at all. We now compare these one-level methods with our multi-level deformation ("mult lev") with $i_{\min} = 3$, $i_{\text{incr}} = 1$ and $N_{\text{pre}} = 2$. The results (figure 10) indicate that *the multilevel deformation features first order convergence*,

i.e. $Q_0 = \mathcal{O}(h)$ and $Q_\infty = \mathcal{O}(h)$. Moreover, *the quality measures are comparable with OneLevConv* for all investigated levels of refinement. Runtime comparisons (figure 9) however demonstrate that multilevel deformation is up to 15 times faster.

The accuracy of any of our deformation algorithms is limited by three error sources: the consistency error, the error induced by solving equation 3 numerically and the error resulting from the approximate solution of all IVPs. All three errors must thus decrease with grid refinement due to the convergence observed. For the consistency error being $\mathcal{O}(h)$, this is obvious. The same holds true for the error of the solution of the deformation PDE. The somehow surprising fact that apparently the IVP-induced error decreases as well despite the constant time step size requires further investigations.

Multilevel deformation works as there is obviously no substantial difference between deforming first and refining afterwards and vice versa. This makes us interpret the deformation step on an intermediate level as correction step to a pre-deformed grid. We expect that the corrections become smaller the finer the grid is. We denote the grid points of $\mathcal{T}_1$ by x and its counterpart on $\mathcal{T}_2$ by X. Then, we measure the distance of these two grids on level k by

$$d_k := \max_{x \in \mathcal{V}} ||x - X||, \quad d_k^{\text{avg}} := \frac{1}{|\mathcal{V}|} \sum_{x \in \mathcal{V}} ||x - X||.$$

Here, $\mathcal{V}$ stands for the set of vertices. Denoting the mapping constructed by the deformation algorithm on refinement level k by Φ_k, we obtain $||x - \Phi_k(x)|| \leq d_k \; \forall \; x \in \mathcal{V}$. Let us furthermore assume that $d_k = \mathcal{O}(h^{1+\tau}).\tau > 0$. We denote the image of the grid point x by X_h, if it was obtained by solving the PDE exactly, but the corresponding IVP approximately. The image of x computed by solving both differential equations with numerical methods we refer to as $\tilde{X}$. We moreover assume that the *relative error induced by the approximate IVP solve* is bounded, i.e.

$$\frac{||X_h - \tilde{X}||}{||x - X||} \leq C \quad \forall x \in \mathcal{V}.$$

This is a by far weaker condition on the IVP solve than the true convergence we required up to now. Then, it follows $||X_h - \tilde{X}|| = \mathcal{O}(h^{1+\tau})$ and the IVP-induced error decreases as $h^{1+\tau}$ *without increasing the number of IVP time steps.* Together with the decay of the other two error sources, this explains the convergence observed. For test problem 1, numerical experiments show that $d_k = \mathcal{O}(h^2)$ (figure 11). We summarize our results in the following conjecture.

Conjecture 1. *Let the sequence of initial grids $(\mathcal{T}_i)_{i \in I}$ emerge from regular refinement of one coarse grid and assume $0 < \varepsilon < f \in \mathcal{C}^1(\bar{\Omega})$ and $0 < g_{\min} < g < g_{\max}$. Then, N denoting the number of grid points, the multilevel deformation algorithm*

a) is of optimal asymptotic complexity $\mathcal{O}(N)$

b) converges with first order: $Q_0 = \mathcal{O}(h)$, $Q_\infty = \mathcal{O}(h)$.

6 Applications

In their investigation on rigid particaluate flows, Turek and Wan [20] considered among other examples the numerical simulation of the sedimentation of many balls with same

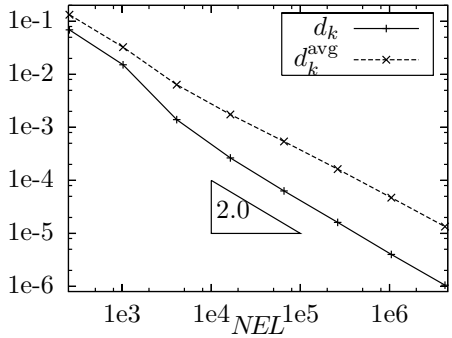

Figure 11: d_k and d_k^{avg} vs. *NEL*, test problem 1

size using multilevel deformation combined with the fictitious boundary method (FBM). The flow field was computed by a special ALE formulation of the Navier-Stokes equations to compensate the mesh deformation in time. The equations were discretised by non-conforming Finite Elements, the resulting linear subproblems were solved with multigrid techniques in the framework of the software package FeatFlow [3]. Solid particles could move freely through the computational mesh which dynamically concentrated around the falling balls by grid deformation using the distance to the particles as monitor function. Numerical results showed the benefit of grid deformation in particulate flows with many moving rigid particles. This work inspired the following test problem.

Test Problem 2. *On* $\Omega = [0, 1.5] \times [0.6]$*, we denote for a ball with index* i*, center* $(x_{c,i}, y_{c,i})$ *and radius* r_i *the euclidean distance of* (x, y) *to* $(x_{c,i}, y_{c,i})$ *by* $d_i(x, y)$*. With* $f_i(x, y) := \max\{(d_i(x, y) - r_i)/r_i, \varepsilon\}$*, we define the global monitor function:*

$$f_{\text{mon}}(x, y) := \left(\Pi_{i=1}^{4} f_i(x, y)\right)^{1/2}.$$

We consider four balls with $r_i = 0.25$ *and centers* $(1, 1)$*,* $(1.55, 1.1)$*,* $(4, 0.3)$ *and* $(3.9, 1.1)$*, respectively and set* $\varepsilon = 0.01$*. A corresponding grid is displayed in figure 12.*

We compute this test problem with our multilevel deformation on an equidistant tensor product mesh. We set $i_{\text{incr}} = 1$ and $i_{\min} = 3$ leading to a coarse grid with 256 elements. We apply RK3 with equidistant step size 0.2 as ODE solver and solve the deformation PDE (3) with the multigrid method described above. However, we now stop the iteration when the residuum is decreased by a factor of 10^6. Before every refinenement step and after every one-level deformation, we employ two steps of Laplacian smoothing. Both the quality measures (figure 13, left) and the runtime results (figure 13, right) confirm conjecture 1.

7 Conclusions and Outlook

We presented a novel method for grid deformation which is based on the basic grid deformation we recently introduced [11, 12]. By subdividing the deformation problem in a sequence of easier deformation problems, we significantly improved the robustness of the grid deformation method. Exploiting grid hierarchy leads to a *multilevel grid deformation*

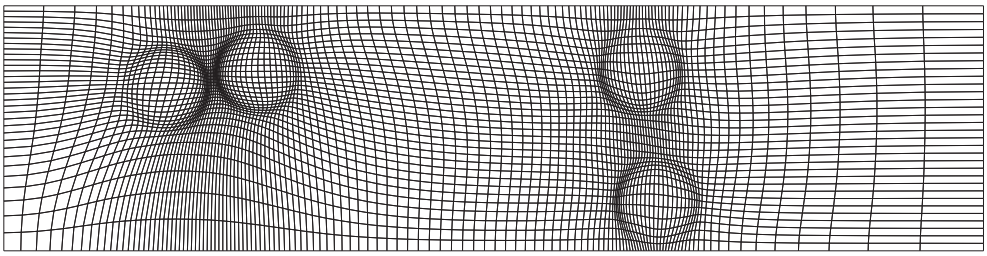

Figure 12: Resulting grid for test problem 2

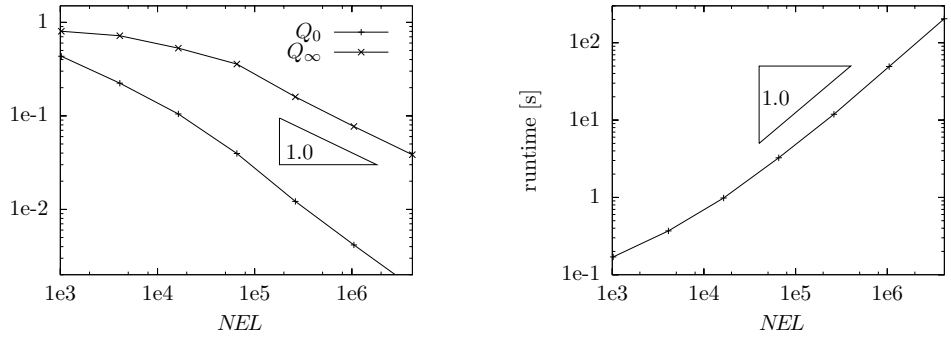

Figure 13: Quality measures (left) and total deformation runtime (right) for test problem 2 computed with multilevel deformation

method converging with optimal order. Moreover, its runtime grows linearly with the problem size. These properties make our new algorithm suitable for applications in r- and rh-adaptive algorithms for simulating challenging real world problems. First numerical results for both static and time-dependent problems show encouraging improvements of the accuracy of the computations. For details, we refer to a forthcoming paper [13] which is devoted to r- and rh-adaptive methods based on the multilevel grid deformation algorithm presented here.

Acknowledgements

This work is partially supported by the German Research Association (DFG) through the collaborative research center SFB 708 and through the grant TU 102/24-1 (SPP 1253). We thank M. Möller for his valuable suggestions regarding this article. Thanks to the FEAST group (C. Becker, S. Buijssen, D. Göddeke, M. Grajewski, H. Wobker) for developing FEAST and the corresponding infrastructure.

References

[1] T. Apel, S. Grosman, P. Jimack, and A. Meyer. A new methodology for anisotropic mesh refinement based upon error gradients. *Appl. Numer. Math.*, 50(3-4):329–341, 2004.

[2] Ch. Becker, S. Kilian, and S. Turek. Hardware-oriented numerics and concepts for PDE software. In *FUTURE 1095*, pages 1–23. Elsevier, 2003. International Conference on Computational Science ICCS2002, Amsterdam.

[3] Ch. Becker and S. Turek. Featflow - finite element software for the incompressible Navier-Stokes equations. User manual, Universität Dortmund, `http://www.featflow.de`, 1999.

[4] P. B. Bochev, G. Liao, and G. C. de la Pena. Analysis and computation of adaptive moving grids by deformation. *Numerical Methods for Partial Differential Equations*, 12:489ff, 1996.

[5] J. U. Brackbill and J. S. Saltzman. Adaptive zoning for singular problems in two dimensions. *J. Comput. Phys.*, 46:342–368, 1982.

[6] X.-X. Cai, D. Fleitas, B. Jiang, and G. Liao. Adaptive grid generation based on least-squares finite-element method. *Computers and Mathematics with Applications*, 48(7-8):1077–1086, 2004.

[7] W. Cao, W. Huang, and R. D. Russell. A study of monitor functions for two-dimensional adaptive mesh generation. *SIAM Journal on Scientific Computing*, 20(6):1978–1994, 1999.

[8] G. F. Carey. *Computational Grids: Generation, Adaptation, and Solution Strategies*. Taylor und Francis, 1997.

[9] B. Dacorogna and J. Moser. On a partial differential equation involving Jacobian determinant. *Annales de le Institut Henri Poincaré*, 7:1–26, 1990.

[10] L. Formaggia and S. Perotto. Anisotropic error estimates for elliptic problems. *Numerische Mathematik*, 94:67–92, 2003. DOI 10.1007/s00211-002-0415-z.

[11] M. Grajewski. *A new fast and accurate grid deformation method for r-adaptivity in the context of high performance computing*. PhD thesis, TU Dortmund, March 2008. `http://www.logos-verlag.de/cgi-bin/buch?isbn=1903`.

[12] M. Grajewski, M. Köster, and S. Turek. Mathematical and numerical analysis of a robust and efficient grid deformation method in the finite element context. submitted to SIAM Journal on Scientific Computing, Dortmund University of Technology, Vogelpothsweg 87, 44227 Dortmund, 2008.

[13] M. Grajewski and S. Turek. Establishing a new grid deformation method as tool in *r*- and *rh*-adaptivity. in preparation, Dortmund University of Technology, Vogelpothsweg 87, 44227 Dortmund, 2008.

[14] G. Liao and D. Anderson. A new approach to grid generation. *Applicable Analysis*, 44:285–298, 1992.

[15] F. Liu, S. Ji, and G. Liao. An adaptive grid method and its application to steady Euler flow calculations. *SIAM Journal on Scientific Computing*, 20(3):811–825, 1998.

[16] Miemczyk. M. Hexaeder-Gittergenerierung durch Kombination von Gitterdeformations-, Randadaptions- und "Fictitious-Boundary"-Techniken zur Strömungssimulation um komplexe dreidimensionale Objekte. `http://www.mathematik.uni-dortmund.de/lsiii/static/schriften_ger.html`. Diploma thesis, Dortmund University of Technology, 2007.

[17] R. Panduranga. Numerical analysis of new grid deformation method in three-dimensional finite element applications. `http://www.mathematik.uni-dortmund.de/lsiii/static/schriften_ger.html`. Master thesis, Dortmund University of Technology, 2006.

[18] S. Turek, D. Göddeke, Ch. Becker, S. H. M. Buijssen, and H. Wobker. UCHPC - UnConventional High Performance Computing for Finite Element Simulations. Technical report, FB Mathematik, Universität Dortmund, February 2008. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 360, submitted to CCPE.

[19] S. Turek, A. Runge, and Ch. Becker. The FEAST indices - realistic evaluation of modern software components and processor technologies. *Computers and Mathematics with Applications*, 41:1431–1464, 2001.

[20] D. Wan and S. Turek. Fictitious boundary and moving mesh methods for the numerical simulation of rigid particulate flows. *J. Comput. Phys.*, (22):28–56, 2006.

[21] Z. Zhang and A. Naga. Validation of the a posteriori error estimator based on polynomial preserving recovery for linear elements. *Int. J. Numer. Methods Eng.*, 61(11):1860–1893, 2004.

[22] O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *Int. J. Numer. Methods Eng.*, 33:1331–1364, 1992.