Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik

h–p–Multiresolution Visualization of Adaptive Finite Element Simulations

Bernard Haasdonk, Mario Ohlberger,
Martin Rumpf, Alfred Schmidt,
Kunibert G. Siebert

Report 01–12

Berichte aus der Technomathematik

Report 01–12 Oktober 2001

h–p–MULTIRESOLUTION VISUALIZATION OF ADAPTIVE FINITE ELEMENT SIMULATIONS

BERNARD HAASDONK, MARIO OHLBERGER, MARTIN RUMPF, ALFRED SCHMIDT, AND KUNIBERT G. SIEBERT

Abstract. We propose an appropriate and efficient multiresolution visualization method for so called h–p–adaptive finite element data. This is done by generalizing the method of adaptive projection, recently introduced by the authors, to function data of piecewise higher polynomial degree on locally refined grids. Given some suitable error indicators, we efficiently extract a continuous h–p–adaptive projection with respect to any prescribed threshold value for the visual error. This projection can then be processed by various local rendering methods, e.g. color coding of data or isosurface extraction. Especially for color coding purposes modern texture capabilities are used to directly render higher polynomial data by superposition of polynomial basis function textures and final color look up tables. Numerical experiments from CFD clearly demonstrate the applicability and efficiency of our approach.

Introduction

Nowadays numerical methods for the simulation of continuum mechanical phenomena continuously advance in their performance and algorithmical complexity. Finite element methods provide a widely used tool for the solution of problems with an underlying variational structure. Modern numerical analysis and implementations for finite elements provide more and more tools for the efficient solution of even large-scale applications. Efficiency can be increased by using local mesh adaption, higher order elements, where applicable, and by fast solvers. Adaptive procedures for the numerical solution of partial differential equations started in the late 70's and are now standard tools in science and engineering. Adaptive finite element methods are a meaningful approach for handling multi-scale phenomena and making realistic computations feasible, especially in 3D.

The task of visualizing interesting solution features of 2D and 3D data sets, as they come along with such adaptive simulations, is thus increasingly demanding, as datasets are usually very large and come along with complex data structures describing local function spaces beyond the simplest linear case. Nevertheless interactivity in post-processing is indispensable for effective and efficient post processing and analysis of given data. Multiresolutional techniques have proved to be very powerful tools in that respect. If numerical data is already characterized by its hierarchical structure a corresponding visualization method is required to make use of this structure for post processing purposes too. In [25, 26] a general concept for the visualization of multilinear finite element data has been presented.

Date: September 21, 2001.

1991 *Mathematics Subject Classification.* Primary 76M27, 65N30, 65S05.

Key words and phrases. multiresolution visualization, higher order, adaptive methods, error indicator, finite element.

Grid cells are supposed to be tensor products of simplices, recursively refined via arbitrary refinement. In [28] this approach has been generalized conceptually to arbitrary nested function spaces. The essential ingredient of the approach is a suitable saturation condition for error indicators corresponding to the available degrees of freedom. This saturation conditions turns out to be fairly natural and can be ensured by a preprocessing of data. Basically, the method extracts an adaptive projection of the finite element function in a depth first traversal of the grid hierarchy. This projection can then be forwarded to any local rendering tool on grid cells.

Here, we will focus on the increasingly important class of h–p–adaptive finite element data and derive an efficient multiresolutional algorithm to extract an adaptive projection on suitable grid levels and of suitable local polynomial degree. Hence, local error indicators are not only given with respect to the grid cells or grid nodes but also for the different possible polynomial degrees. A threshold which is prescribed by the user controls which grid level and which polynomial degrees have to be selected locally. As in the numerical h–p–method itself we assume that higher polynomial degrees show up on the locally finest grid level only. With different choices of the threshold, different situations occur, which will be illustrated more detailed in Section 3. If the threshold value is large enough, the extraction is h–adaptive only. Below some critical value, the extraction is purely p–adaptive, all elements are geometrically refined up to the leaf level and a proper choice for the local degree of the polynomial approximation is required. Finally, for threshold values in between the adaptive projection will be piecewise linear on coarse cells in some region of the domain and resolved by higher polynomial degree in other regions. Obviously, ensuring global continuity is the crucial quality criterion.

Once having calculated the adaptive projection, some local rendering tool on the grid cells comes into play. The most frequently used tool in 2D is the color shading of a scalar function directly on the cells and in 3D the color shading on a slice through the cell. Here, texture hardware will enable us to compute the required linear combination of polynomial basis functions on each cell or cell slice and finally color code the resulting polynomial due to user prescribed parameters. Neglecting the computational cost in the texture hardware the local computing cost of the method on a single cell grows linear with the number of involved base functions. Thus the performance of the method is of optimal order.

Related work. A variety of applications, such as terrain visualization, surface modeling, medical imaging and especially numerical simulations deliver enormous amounts of data. Multiresolutional techniques have already proven to be the adequate solution for a large class of applications to allow an interactive exploration of data sets. Various authors have approached them in multiple ways. We give a brief and naturally incomplete overview.

The efficient rendering of height fields, in geographic imaging especially for flight simulation purposes has been studied e. g. by Certain et al. [8], Faust et al. [13], Floriani et al. [12], and Gross and Staadt [16].

For arbitrary triangular surfaces, e. g. isosurfaces in numerical data fields, surfaces generated by some 3D scanning process, or shapes in geometric modeling, adaptive coarsening strategies have been presented by Turk [38], Hamann [18], Schroeder et al. [33], Hoppe [20] and others. For a comparison of different mesh simplification algorithms we refer to Cignoni et al. [11].

A fast and adaptive visualization of volume data is implemented in the hierarchical splatting algorithm by Laur and Hanrahan [22]. Cignoni et al. [10] have applied a successive adaptive refinement of volumes by Delaunay methods. Adaptive isosurface extraction has for instance been treated by Shekhar et al. [34], Zhou et al. [41], and Grosso et al.[17] with different approaches to solve the outstanding continuity problem, e. g. to avoid cracks in adaptive isosurfaces.

Various techniques have been presented to use texture hardware to speed up the post processing of scientific data. Westermann and Ertl [39] made use of texture hardware for high quality splatting in volume rendering. Lippert and Gross [23] used wavelet splats to directly render wavelet compressed volume data. Already in 1992 Zumbusch [42] used polynomial textures for color shading. More advanced usage of the arithmetic capabilities of texture hardware have been considered with respect to image convolution [19] and image processing [37, 36].

Organization of the paper. In the next section we will introduce some notations and give a rigorous definition of h– and p–hierarchy before we build up the concept of h–p–multiresolution visualization in Section 2. For the ease of presentation we will confine ourselves to simplicial grids throughout the paper. Nevertheless, based on the more general concepts presented in [25, 26] the generalization to hierarchical grids, the cells of which are tensor products of simplices, is straightforward. Concerning the rendering of polynomial functions on simplices we introduce polynomial textures, based on a composition of basis textures in Section 3. In Section 4 we will then briefly review higher order finite element methods on adaptive grids and describe the basic design principles for an implementation. Thereby we especially emphasize that the proposed multiresolutional approach nicely corresponds to a suitable h–p finite element solution strategy. Section 5 discusses the interface between visualization and numerics. Finally, in Section 6 two applications from CFD relying on a higher order discretization are visualized by the proposed multiresolution techniques.

1. Notations and h–p–hierarchies

In the following section we will discuss hierarchical data structures as they are actually used in the h–p–adaptive finite element code. We will deal with simplicial grids in 2D and 3D simultaneously and only for the final local rendering we have to outline the dimensional peculiarities. In what follows we suppose data sets are defined on a hierarchically nested simplicial grid as piecewise polynomial data given on the finest grid level. Furthermore we assume that the given data function is globally continuous on the finest level. We will refer to the grid hierarchy as h–hierarchy and introduce an additional p–hierarchy on every leaf element by a suitable interpolation of the piecewise polynomial data onto the space of polynomials of degree p for increasing values of p. Let us now briefly introduce some notations and definitions.

1.1. ***h*–hierarchy.** We start with a brief overview of nested spatial subdivisions. Let Σ^n be the set of closed simplices of dimension n, e. g. Σ^0 denotes vertices, Σ^1 edges, Σ^2 triangles, and Σ^3 tetrahedrons. We consider grids in dimension $n = 2, 3$ consisting of elements $E \in \Sigma^n$, $E \subset \mathbb{R}^n$. The boundary of every element splits into $n - 1$ dimensional sub–simplices from Σ^{n-1}. These sub–simplices share at their boundary further sub–simplices from Σ^{n-2} with other sub–simplices. Thereby, in

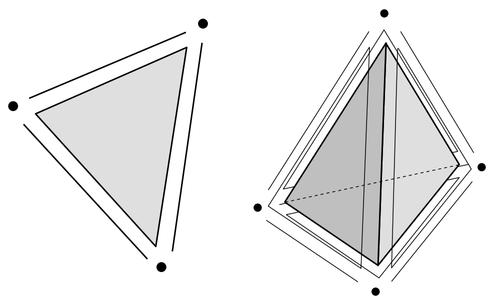

FIGURE 1. Relation between higher and lower dimensional simplices as boundary simplices.

2D we have already reached the level of vertices, whereas in 3D a further level is required to end up with the zero dimensional simplices corresponding to the nodes of the grid, cf. Figure 1.

A conforming mesh $\mathcal{M}$ for a domain $\Omega \subset \mathbb{R}^n$ is a set of closed simplicial elements E such that $\bigcup_{E\in\mathcal{M}} E = \Omega$ and any two elements of $\mathcal{M}$ are disjoint or intersect in some common boundary simplex, e. g. a face, edge or vertex. A family of conforming meshes $\{\mathcal{M}^l\}_{0\le l\le l_{\max}}$ is called a nested grid, if for all $E^{l+1} \in \mathcal{M}^{l+1}$ there exists an $E^l \in \mathcal{M}^l$ with $E^{l+1} \cap E^l = E^{l+1}$. We suppose these grids to be recursively generated by a finite set of refinement rules applied to certain elements of the preceding, coarser mesh. Those elements $E^{l+1} \in \mathcal{M}^{l+1}$, generated from $E^l \in \mathcal{M}^l$ by subdivision, will be called child elements $\mathcal{C}(E^l)$ of element E^l. Vice versa the parent relation $\mathcal{P}(E^{l+1}) {:=} E^l$ holds for every $E^{l+1} \in \mathcal{C}(E^l)$. We assume that every child element $E^{l+1} \in C(E^l)$ intersects the boundary of its parent element E^l.

1.2. ***p*–hierarchy.** In contrast to the h–hierarchy, the p–hierarchy is not deduced from a spatial subdivision of grid elements, but from the interpolation of given data into piecewise polynomial function spaces of degree p. For simplicity let us suppose that the function f on an element E is given by a polynomial of degree $p(E)$, i.e. $f \in \mathbb{P}(E, p(E))$, where $\mathbb{P}(E,p)$ is the set of polynomials of degree p on element E. We suppose $\mathcal{L}(E,p) := \{x_i^p |\ i \in I(p)\}$ to be the set of Lagrange nodes (cf. Fig. 2) and $V(E,p) := \{\varphi_i^p |\ i \in I(p)\}$ with $\varphi_i^p \in \mathbb{P}(E,p)$ and $\varphi_i^p(x_j^p) := \delta_{ij}; i,j \in I(p)$ to be the corresponding set of Lagrange basis functions. For details we refer to [9]. Thus, we have $\mathbb{P}(E,p) = \operatorname{span} V(E,p)$, i.e. for all $f \in \mathbb{P}(E,p(E))$ we can write

$$f(x) = \sum_{i\in I(p(E))} f(x_i^{p(E)})\varphi_i^{p(E)}(x).$$

For such a polynomial f on a simplex E let us now define a hierarchy of representations by the following interpolations I_q

$$I_q(f) := \sum_{i\in I(q)} f(x_i^q)\varphi_i^q(x). \tag{1}$$

This hierarchy of representations for all $1 \le q \le p(E)$ will be called p–hierarchy.

1.3. **Corresponding hierarchy of function spaces.** Corresponding to the h–hierarchy on the conforming nested grids $\{\mathcal{M}^l\}_{0\le l\le l_{\max}}$ we consider a family of

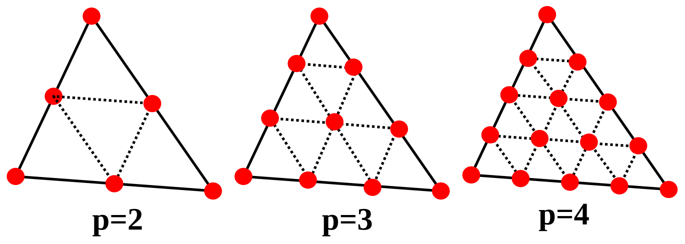

FIGURE 2. The sets of Lagrange nodes $\mathcal{L}(E,p)$ for increasing polynomial degrees p.

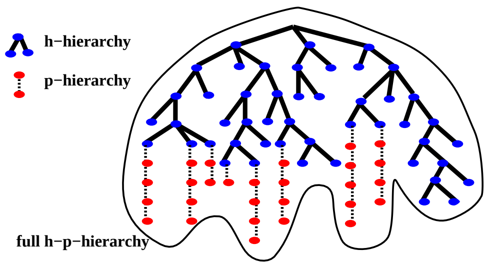

FIGURE 3. The sketch of an h–p–hierarchy, where blue nodes correspond to hierarchical cells, red nodes to different polynomial degrees on a cell and the solid and dotted lines emphasize the hierarchical relations.

discrete piecewise linear finite element function spaces $\{\mathcal{V}^l\}_{0\le l\le l_{\max}}$, which are ordered by set inclusion:

$$\mathcal{V}^0 \subset \mathcal{V}^1 \subset \cdots \subset \mathcal{V}^l \subset \mathcal{V}^{l+1} \subset \cdots \subset \mathcal{V}^{l_{\max}}.$$

This hierarchy of spaces can be extended by increasing the polynomial degree of the functions on leaf elements $E^{l_{\max}} \in \mathcal{M}^{l_{\max}}$. Thus we obtain the h–p–hierarchy

$$\mathcal{V}^0 \subset \mathcal{V}^1 \subset \cdots \subset \mathcal{V}^{l_{\max}} = \mathcal{V}_1^{l_{\max}} \subset \mathcal{V}_2^{l_{\max}} \subset \cdots \subset \mathcal{V}_{p_{\max}}^{l_{\max}}.$$

Here $\mathcal{V}_p^{l_{\max}}$ denotes the function space of globally continuous functions, which are in $\mathbb{P}(E,p)$ on every leaf element E of the mesh. Figure 3 illustrates the complete hierarchy.

To ensure globally conforming function data the polynomial degree on boundary sub–simplices (faces, edges, vertices) shared by different simplicial cells has to be properly adjusted. Due to our assumption on the continuity of the given function we assume this property to hold on the finest h and p level for given data. It will be the key objective of our considerations to enforce this condition also for our adaptive projection which is computed based on some error indicators supplied by the user. Thus, in the case of higher order finite elements we have a given h–hierarchy by the conforming nested grid and an additional p–hierarchy by the interpolations I_q defined in equation (1) on each leaf element.

1.4. **Further useful notations.** Different from the vertex based notation chosen in [25, 26] we will focus here on a more appropriate simplex based notation. Vertices appear as 0 dimensional simplices. We will indicate the highest dimensional simplices usually by E, whereas simplices of arbitrary dimension will be denoted by e. For $e \in \Sigma^d$ we denote by $\partial e \subset \Sigma^{d-1}$ the set of boundary simplices of e. Furthermore, for any simplex e, $\mathcal{C}(e)$ is defined as the set of child simplices of the same dimensionality generated by the h–refinement of the corresponding cell. The reverse parent relation is expressed in terms of a parent simplex $\mathcal{P}(e)$. Most boundary simplices can be regarded to be part of several higher dimensional simplices. However, our definition of the child sets of boundary simplices of any dimension is well–posed as the different child sets coincide due to our claim for mesh conformity.

2. h–p–MULTIRESOLUTION VISUALIZATION

Before we describe the h–p–adaptive concept as a natural extension of the pure h–adaptive approach let us briefly sketch the case of hierarchical piecewise linear function spaces.

2.1. A review on h–multiresolution visualization.

Let us suppose that a spatial hierarchy is given by conforming nested simplicial grids as they have been introduced in Section 1. Here we use a slightly different notation compared to the one used in [26]. It will turn out to be well–suited for its later generalization to the h–p–context. The aims are:

- to apply local coarsening in a simple depth first traversal of the grid hierarchy,
- to use error indicator values which steer the selecting process of suitable local grid levels,
- to consider a continuous restriction of a data function given on the finest grid level to the adaptively extracted coarse grid,
- and finally to ensure certain smoothness of the grid resolution.

As described in [26] this can be achieved by a stopping criterion defined on grid cells and depending on certain error indicators, a saturation condition ensuring the suitable transport of error information on fine grid cells to coarse grid cells, and an induced recursive definition of an adaptive projection $P_{\mathcal{S}}U$ of some data function U. Here, we consider this process to be steered by error indicator values $\eta(e)$ on simplices e. The recursive traversal of the grid hierarchy is stopped wherever the criterion

$$\mathcal{S}(e) := (\eta(e) \leq \varepsilon)$$

is fulfilled for some user prescribed threshold value ε on some simplex $E \in \Sigma^n$. For simplices E this criterion is therefore called *stopping* criterion, whereas for lower dimensional simplices the notion *projection* criterion is used. We impose the following saturation conditions on the error indicators:

(h–saturation condition) *For all simplices e we require*
(i) $\eta(e) \geq \eta(e_\partial)$ for all $e_\partial \in \partial e$,
(ii) $\eta(e) \geq \eta(e_c)$ for all $e_c \in \mathcal{C}(e)$, or
(ii') $\eta(e_\partial) \geq \eta(e_c)$ for all $e_c \in \mathcal{C}(e)$ and all $e_\partial \in \partial e$ and $e_c \cap e_\partial \neq \emptyset$.

The first condition (i) ensures continuity, the second one (ii) requires error indicators on coarser grid levels to dominate those on descendent simplices. Alternatively we may state condition (ii') to ensure a suitable homogeneity of the grid structure, on which our adaptive projection is finally defined. Obviously conditions (i) and (ii') imply (ii). One easily verifies [26] that there is at most one grid level difference

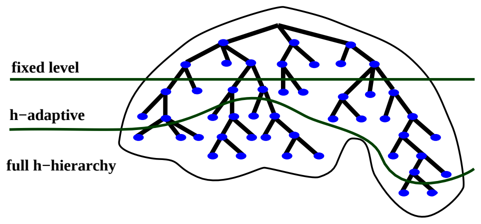

FIGURE 4. A sketch of a pure h–adaptive traversal

between elements sharing a common face in the resulting adaptive grid. The adaptive projection is obtained as follows. If the projection criterion $\mathcal{S}(e)$ is fulfilled on some boundary simplex e and in the adaptive traversal of the grid this simplex has to be refined further, we replace the function on the next finer level by the restriction to the current level. Because linear functions on simplices are uniquely defined by linear functions on their boundary simplices this leads to a piecewise linear adaptive projection $P_{\mathcal{S}}U$ and ensures its continuity.

In fact we can confine to error indicator values given on grid nodes only. I. e. we define

$$\eta(e) := \max_{x\in\mathcal{N}(e)} \eta(x)$$

where $\mathcal{N}(e)$ is the set of vertices of e. Thus the saturation condition reduces to the single condition

$$\eta(x) \geq \eta(x_c), \qquad \text{for all } \; x_c \in \mathcal{N}(E_c) \setminus \mathcal{N}(E)$$

for nodes $x \in \mathcal{N}(E)$ of all grid cells E and all child elements $E_c \in \mathcal{C}(E)$.

The procedure which uniquely defines the adaptive projection $P_{\mathcal{S}}U$ also simplifies: If during the recursive grid traversal a new vertex appears and the corresponding error indicator is below the threshold ε, we use linear interpolation to compute the nodal value of the adaptive projection. Otherwise we retrieve the true function value. Finally, in the traversal we stop on an element E if the error indicator values on all cell nodes are less than the threshold value (cf. Fig. 4).

For more details on this multiresolution approach, possible choices of error indicators and an efficient method to guarantee the saturation condition we refer to [26].

2.2. *p*– and *h*–*p*–multiresolution visualization.

In the following, we generalize the concept of multiresolution visualization for piecewise polynomial data. Some preliminary ideas have been proposed in [26]. Here the scope is more general, as it enables us to use other than piecewise linear drawing methods. To be precise, we can use any adequate visualization method for polynomial data. For example NURBS [27] can be applied for function-graph drawing on 2D meshes and on slices through 3D cells, respectively. Below we will focus on color shading based on polynomial textures (cf. Section 3).

The p–multiresolution approach. To prepare the general discussion of h–p–adaptive strategy we first discuss the control of the polynomial degree on finest level simplices of a homogeneously refined grid. Instead of defining single error indicators on the set of simplices, as we have done in the pure h–adaptive case, we now define on each

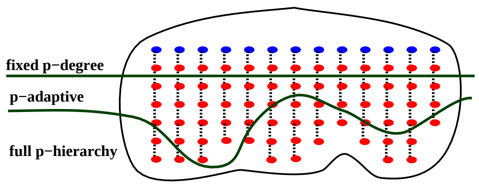

FIGURE 5. p–multiresolution

simplex e error indicators $\eta_p(e)$ for all possible polynomial degrees p. Analogous to the h–adaptive case, on any simplex we will confine with the maximal polynomial degree for which the error indicator is still above the user given threshold ε and resolve the data function up to this degree. In order to ensure the thereby defined adaptive projection to be continuous, we once more have to enforce a suitable saturation condition, which now reads as follows:

(p–saturation condition) *For all simplices e we require*
(i) $\eta_p(e) \geq \eta_p(e_\partial)$ for all $e_\partial \in \partial e$ and all $p \leq p_{\max}$,
(ii) $\eta_p(e) \geq \eta_q(e)$ for all $p \leq q \leq p_{\max}$, or
(ii') $\eta_p(e_\partial) \geq \eta_{p+1}(e)$ for all $e_\partial \in \partial e$ and and all $p \leq p_{\max} - 1$.

Like in the pure h–hierarchical saturation condition, the alternative condition (ii') is solely required, if a suitable homogeneity of the polynomial degrees should be achieved. Indeed, if we assume the meaningful condition $\eta_p(e^0) = \eta_q(e^0)$ for $e^0 \in \Sigma^0$, then (i) and (ii') again imply (ii). Due to the subset relation of the simplices and the asymptotically exponential convergence for increasing polynomial degree and smooth data, this saturation condition turns out to be a natural condition. Furthermore, we define a stopping criterion on the simplices depending on the user prescribed error bound ε and the polynomial degree q:

$$\mathcal{S}_q(e) := (\eta_q(e) \leq \varepsilon)$$

Then the corresponding visualization algorithm proceeds as follows: The p–adaptive projection on each simplex e is determined choosing the smallest possible polynomial degree $q(e) \leq p_{\max}$, such that the stopping criterion $\mathcal{S}_{q(e)}(e)$ is true. In fact our adaptive projection algorithm is cell oriented and we consider an adaptive projection of degree $q(E)$ on a cell E. To respect the probably lower – due to the saturation condition definitely not larger – polynomial degree of any descendent boundary simplex e we choose in advance proper values at those Lagrange nodes with respect to the cell E, which partially lie on its boundary simplices. I. e. for a Lagrange node x on a boundary simplex e we choose $(I_{q(e)}(f))(x)$. Due to this construction we obtain a globally continuous adaptive projection.

Let us emphasize, that for two dimensional simplicial grids error indicators have to be stored on edges and elements, whereas in the three dimensional case we have to store indicators on edges, faces and elements. One additional error indicator on each vertex is only necessary, if we consider h–adaptivity as well (see below).

The full h–p–multiresolution approach. So far we have introduced the multiresolution algorithm in the case of a pure h-hierarchy or pure p-hierarchy. Now we combine these two concepts in a straightforward way. Here we especially benefit from reformulation of the h–adaptive method in terms of error indicators given on all simplices of any dimension, although they are trivially defined by the set of

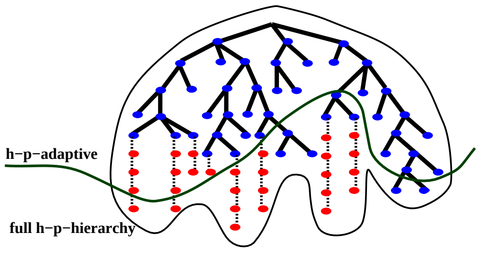

FIGURE 6. A sketch of a h–p–multiresolutional hierarchy.

nodal error indicators. We simply consider the h–error indicators as error indicators corresponding to the polynomial degree 1, i. e.

$$\eta_1(e):=\eta(e)$$

for every simplex e appearing in the simplicial grid. Vertices obviously play a special role: Although indicators of various degree are defined, they are assigned to a single function value. However, this does not influence our indicator concept. With this identification at hand we need no further saturation condition and the proof that the resulting adaptive projection is continuous is left to the reader.

The advantage of this adaptive approach is near at hand. Similar to the pure h–adaptive case we get along with a depth first traversal of the tree hierarchy. If the saturation conditions hold, global continuity of the resulting projection $P_{\mathcal{S}}f$ is always ensured (cf. Fig. 6).

2.3. **Implementational aspects.** So far we presented the theoretical concept. Let us now discuss some relevant aspects concerning the practical application. In particular we comment on the calculation of the error indicators and how to satisfy the saturation conditions.

Choice of visual error indicators. Of course the vague notion of visual error can not be measured in exact mathematical terms as it is determined by human perception and depends on the specific information on which one is focusing in the final image [14]. Therefore, any error indicator has to be some heuristic measure. Measures of discrete curvature are reasonable choices for the nodal indicator values $\eta(x)$ in many cases, e.g. the angle in an appearing crease of an isoline or the jump of the normals across edges. For the error indicators $\eta_q(e)$ on simplices one can use simple L^s-norms for $1 \leq s \leq \infty$ of the approximation error between $I_q(f)$ and f.

Ensuring the saturation conditions. For these particular choices, sufficiently smooth data f and sufficiently fine grid level the h– and p–saturation conditions hold true. In general, on coarser grid levels this is no longer true. Thus, we have to ensure this property by adjusting the error indicator values in a preprocessing step.

The saturation conditions consist of simple hierarchical relations between simplices on different grid levels, of different dimension and corresponding to different polynomial degrees. If we denote by $\Sigma^i(\mathcal{M}^l)$ the set of all simplices $e \in \Sigma^i$ on grid level l, then the saturation algorithm for the h–adaptive method reads as follows:

```
for $l = l_{\max}, \cdots, 0$
    for $i = 0, \cdots, n$
        for all $e \in \Sigma^i(\mathcal{M}^l)$
```

replace $\eta(e)$ by the smallest number larger than
$\eta(e)$ and fulfilling the saturation inequalities:
(i) $\eta(e) \geq \eta(e_\partial)$ for all $e_\partial \in \partial e$,
(ii) $\eta(e) \geq \eta(e_c)$ for all $e_c \in \mathcal{C}(e)$, or
(ii') $\eta(e) \geq \eta(\tilde{e}_c)$ for all $\tilde{e}_c \in \mathcal{C}(\tilde{e})$ and $e \in \partial\tilde{e}$ with
$\tilde{e}_c \cap e \neq \emptyset$.

Here, we choose either condition (ii) or (ii') depending on our choice of the type of saturation condition. A simpler version of this algorithm with respect to the above mentioned nodal based construction can be found in [26]. The saturation with respect to p–adaptivity and the error indicator values $\eta_p(e)$ can be coded analogously exchanging l by p and $l_{\max}$ by $p_{\max}$ and running over all $\eta_p(e)$ with $e \in \Sigma^i(\mathcal{M}^{l_{\max}})$ and $p \geq 1$. Let us emphasize that the relation between the h and p-estimators requires first the p-saturation on the finest grid level followed by the h-saturation on the grid hierarchy.

Complexity. The p–error indicators require additional storage. In 2D the size of this memory demand can be stated exactly depending on the number of elements N, the number of egdes F, the number of nodes M and the polynomial degree p. We need to store $M + (N + F) \cdot p$ indicator values. However, a quantization of the indicators to some prescribed bitlength can be used to reduce the complexity linearly.

3. Extraction with polynomial textures

So far, we described the general approach to visualize polynomial data with edge- and elementwise error indicators. Let us now have a closer look to color shading of the adaptive projection in the two dimensional case. In case of color shading on slices of a 3D simplicial grid we can proceed in an analogous manner. So far, we assumed to have some extraction routine at hand which performs the local rendering of piecewise polynomial data. As a very efficient extraction routine, we now consider polynomial textures (p–textures) to make use of nowadays texture hardware. However, we once more stress that the general concept is also applicable to other methods like polynomial surfaces or polynomial 3D-volume textures.

Notion polynomial texture. The notion polynomial texture in our context is very simple: Texture refers to the usual computer graphics term, i.e. a coloured bitmap which can be pasted on every 2D-object, in our case triangles. *Polynomial texture of degree q* now denotes a bitmap of an 'exact' colorshading representation of some polynomial function f of degree q defined on a triangle E. Such a texture T is uniquely defined by the polynomial function, the size and shape of the bitmap and some colormap $m : \mathbb{R} \to C$ which maps the function values of f to some 3- or 4-dimensional color space C. By the special choice of such a colormap effects like isolines (or isosurfaces in 3D case) can be realized.

Simple examples. We first show basic usage of polynomial textures in Figure 7. We generated some synthetic data containing 3 rotional-symmetric parts on a grid consisting of 16 triangles. The improvement of the representations by increasing the polynomial degree of the p-textures is obvious.

Figure 8 demonstrates the efficiency of fourth order p-textures compared to subsampling with piecewise linear patches. Both images are based on identical

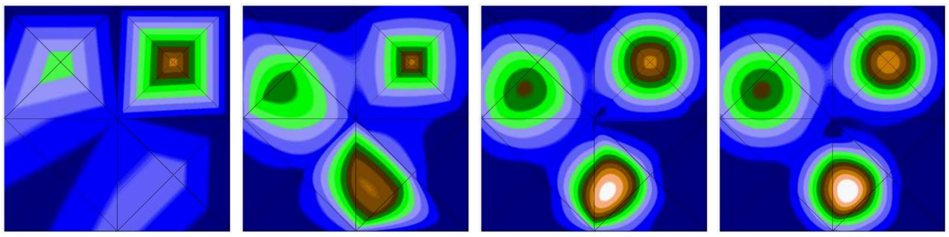

FIGURE 7. Demonstration of improving approximation properties of polynomial textures with degrees increasing from 1 to 4 applied to some synthetic data on a grid consisting of 16 triangles.

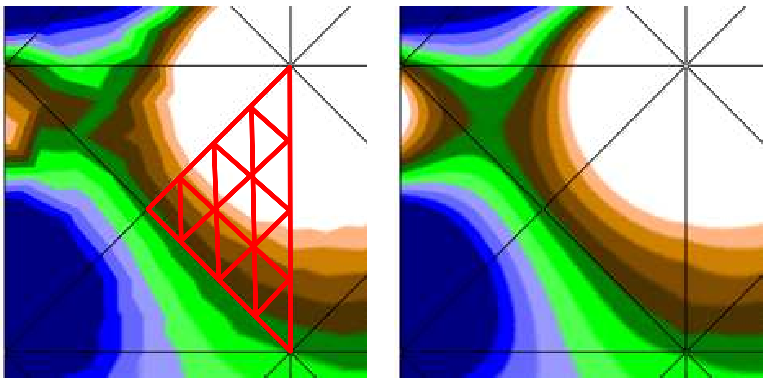

FIGURE 8. Visual comparison of subsampling and use of p–textures on some simple finite element data with fourth order Lagrange-elements. right image: polynomial textures of degree four, left image: straightforward subsampling with linear patches. Black lines indicate the triangular numerical grid, red lines indicate the subsampling level: Each element is refined by two recursive subsampling steps into smaller elements each of which is extracted linearly.

function-evaluations as the nodes of the subsample-grid (indicated in red) coincide with the Lagrange nodes of each triangle.

The visual impression however is much better in the p–texture case, simply due to the fact that p–textures are a natural representation for polynomial data in contrast to any subsampling procedure which only can perform piecewise linear approximations and typically ends up with corners in the image.

3.1. **Implementation.** During visualization many of these p–textures have to be generated. We now describe an efficient way of generating these p–textures which only requires evaluation of the numerical data in the few Lagrange nodes of every triangle and simple linear combination of some basis-textures.

For the unit triangle E_0 (i.e. triangle with corners $(0,0),(1,0),(0,1) \in \mathbb{R}^2$) and each necessary polynomial degree q we sample each of its Lagrange basis functions φ_i^q into some appropriate-sized sample–texture S_i^q. Obviously these textures can be used for visualizing any polynomial function f_0 of degree q on E_0. Recalling that f_0 can be represented by its values $u_i := f_0(x_i^q)$ in the Lagrange nodes $x_i^q \in \mathcal{L}(E_0, q)$ by $f_0(x) = \sum_{i \in I(q)} u_i \varphi_i^q(x)$ for all $x \in E_0$. Therefore we obtain a sample-texture

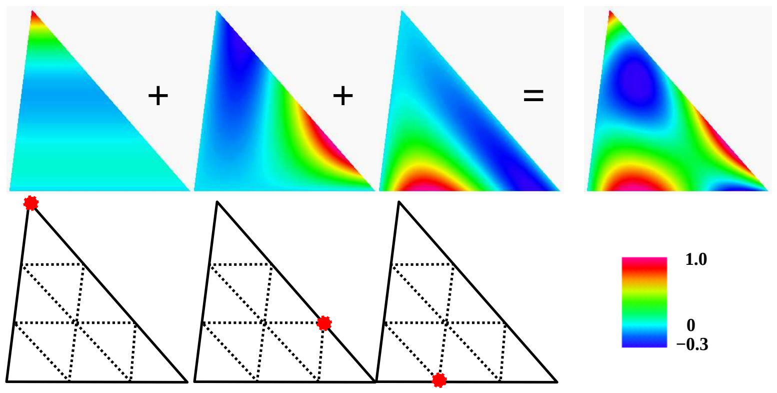

FIGURE 9. Some basis textures S_i^3 mapped to an element E using the indicated colormap. The simple sum of these textures results in a representation of the polynomial function defined by the sum of the corresponding basis-functions $\varphi_i^3 \in V(E,3)$.

S_0 of f_0 by

$$S_0 := \sum_{i \in I(q)} u_i S_i^q. \tag{2}$$

Mapping of this texture with the prescribed colormap m results in a p–texture which can be patched on E_0. Thus, after an affine transformation of E_0, f_0 to E, f we get a perfect representation of f on E. So p–textures for arbitrary polynomial function f on arbitrary triangle E can be constructed by these sample–textures S_i^q (cf. Figure 9).

Note that latest graphics-hardware drivers are capable to perform linear combinations like in (2) within the graphics hardware which gives the necessary acceleration for the efficient rendering.

3.2. **Efficiency tests.**

p–multiresolution visualization. We now want to present some test cases for the different multiresolution approaches. We again choose some finite element simulation data of piecewise fourth order. The geometry consists of a hierarchical mesh covering the unit-square, which is obtained by recursive bisection of four macro triangles. The uniformly refined grid after 10 recursive bisection steps is shown in the topleft image in Figure 10. Data is extracted with p–textures of degree four which is the exact representation and we applied the p–multiresolution algorithm for different tolerances ε. The difference to the exact representation of given data is almost not noticeable although the necessary elementwise polynomial degrees of the textures are decreased in smooth regions as indicated in the topright image. This results in an acceleration of visualization due to less data-evaluations.

h–p–multiresolution visualization. The results in Figure 10 by the p–multiresolution algorithm are identical to the results from the complete h–p–multiresolution algorithm, if ε is so small that all leaf elements are extracted. We now continue the

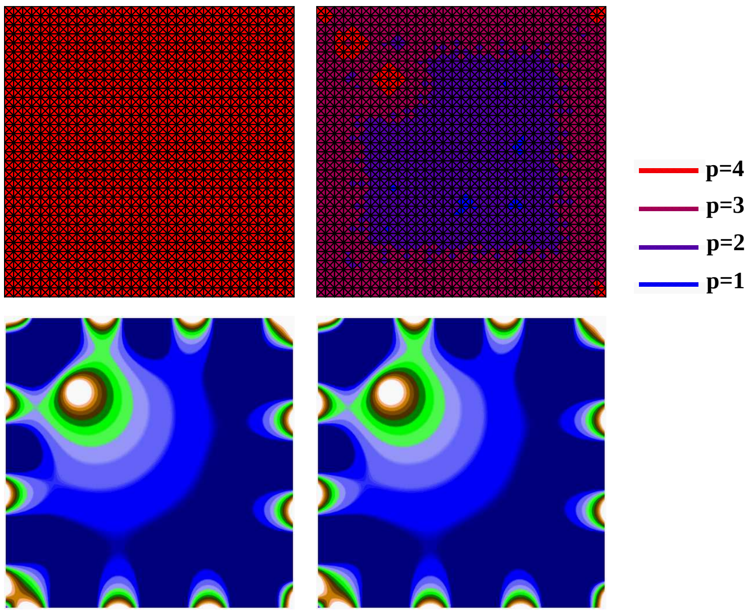

FIGURE 10. Effect of the p–multiresolution approach applied to some 4th order FE-data. Left column: results for $\varepsilon = 0$. Right column: results for $\varepsilon = 0.00005$. Upper row: extraction grid and elementwise polynomial degrees used for visualization. Lower row: result of the extraction with p–textures.

experiments of increasing ε as indicated in Figure 11. The necessary polynomial degrees for visualizing the elements decrease and some elements (indicated as white) now are not completely refined up to the leaf level due to the h-adaptivity.

In the leftmost column all elements are processed linearly and this is the identical result as obtained by the pure h–multiresolution approach.

Complexity. We now give some quantitative arguments for the efficiency of the multiresolution scheme. For each of the 5 different values of ε we determined the required number of evaluations of the data function as this is in general the most expensive operation during the h–p–multiresolution algorithm. These experimental results are given in Figure 12, where the numbers of function evaluations also imply the additional interpolations along edges. The runtime decreases according to the number of function evaluations. Note that this relation is not linear because of a basic overhead by the h–p–algorithm.

Consistency. One aspect which is perfectly satisfied as predicted by the theoretical considerations is the consistency of all results produced by the multiresolution algorithms. We demonstrate this by a magnified detail of the $\varepsilon = 0.002$ case in Figure 13. We see that for all neighbouring leaf elements the edgewise polynomial degrees for the common edge are identical, so the interpolations of given data will be consistent. Furthermore all leaf edges next to a non-leaf element are resolved with polynomial degree 1. Consistency on structured hierarchical meshes is shown in Figures 10, and 11, whereas Figures 19, and 20 also demonstrate the consistency

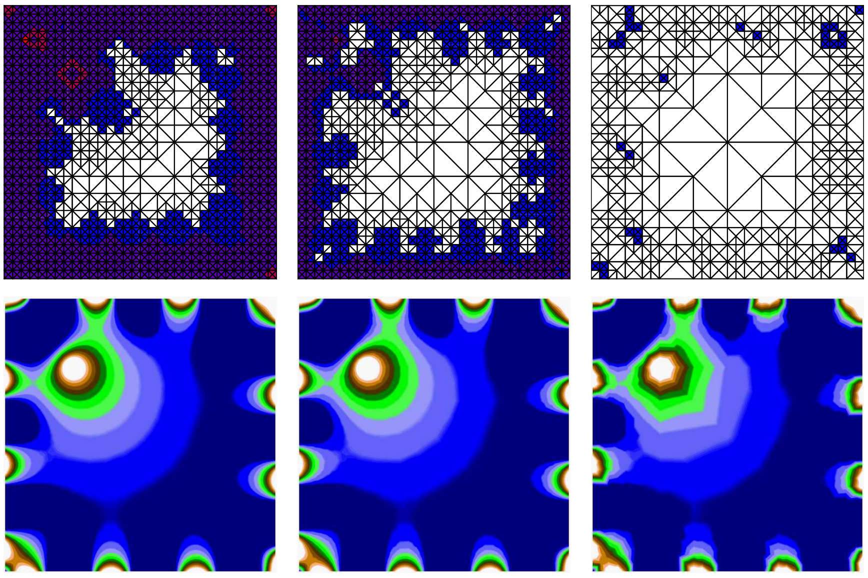

FIGURE 11. Effect of the h–p–multiresolution approach. Columns: results for $\varepsilon = 0.002$, $\varepsilon = 0.01$ and $\varepsilon = 0.1$. Upper row: extraction grids and elementwise polynomial degrees used for visualization. Second row: result of the extraction using p–textures for leaf elements and linear patches for non-leaf elements (indicated as white).

ε	visual impression	number of function eval.	runtime with p–textures
0	uniformly refined, $p = 4$	61440	1.62 sec
0.00005	uniformly refined, $p = 1..4$	36283	1.27 sec
0.002	few coarse elements, $p = 1..4$	16929	0.88 sec
0.01	many coarse elements, $p = 1..3$	11267	0.79 sec
0.1	few leaf elements, $p = 1$	2571	0.67 sec

FIGURE 12. Quantitative results for the complexity gain of the h–p–multiresolution algorithm. The runtimes include the complete h–p–multiresolution-algorithm and linear combinations of the textures.

on unstructured meshes resulting from a subdivision of a Delaunay macro triangulation.

The introduced h–p–multiresolution strategy and the local rendering with p–textures are realised in the programming environment GRAPE [40, 15].

4. Higher order adaptive finite element simulations

Finite element methods calculate approximations to the true solution of partial differential equations in some finite dimensional function space. This space is built from *local function spaces* $\mathbb{P}(E)$, usually polynomials of low order, on elements E of a partitioning $\mathcal{M}$ of the domain (the *mesh*). An adaptive h-method adjusts

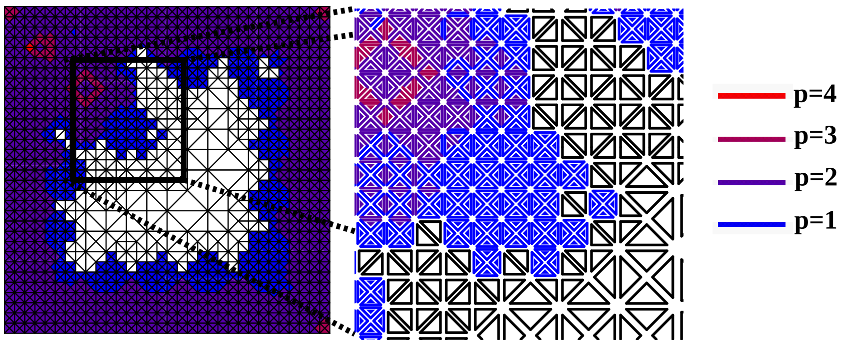

FIGURE 13. Demonstration of the consistency of the h–p–multiresolution algorithm by visualizing the resulting edgewise polynomial degrees.

this mesh to the solution of the problem by refining the mesh locally and thus inserting locally new elements. An adaptive p-method adjusts the local function spaces on selected elements by increasing the dimension of the local spaces. Finally, an adaptive h-p-method combines the h- and p-method by refining the mesh locally in some parts of the domain and enlarging the local function spaces in other parts. Here, we focus on the adaptive h-method. A given mesh is refined locally and the local function spaces are given by *one* function space $\bar{\mathbb{P}}$ on some reference element $\bar{E}$ and the mapping from this reference element to the mesh elements.

Adaptive methods and error estimators. The aim of adaptive methods is the generation of a mesh $\mathcal{M}$ which is adapted to the problem such that a given criterion, like a tolerance for the estimated error between exact and discrete solution, is fulfilled by the finite element solution on this mesh. An optimal mesh should be as coarse as possible while meeting the criterion, in order to save computing time and memory requirements. For time dependent problems, such an adaptive method may include mesh changes in each time step and control of time step sizes.

The adaptation is based on information extracted from *a posteriori error estimators.* These estimators are *computable* error estimates for the error between the true solution and the finite element approximation and they are build up from local *error indicators.* An adaptive method is driven by such an error estimator and tries to optimize the mesh by equidistributing the local indicator values over all mesh elements, while the total estimate is below a given tolerance. Here, a posteriori error estimators provide a proven basis for the adaptive algorithm which result in meshes, which are highly refined only where really needed. There exist a lot of different approaches to (and a large literature about) the derivation of error estimates, by residual techniques, dual techniques, solution of local problems, hierarchical approaches, etc. Here, we just want to give an example for residual estimators, which are also used in the simulations shown later. In contrast to the visual error estimator used for adaptive visualization, which is computed from the function values alone, compare Section 2.3, the equation and data of the continuous problem enter here. For example, for a quasi–linear elliptic problem

$$-\nabla\cdot A\nabla u + f(x,u,\nabla u) = 0 \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega$$

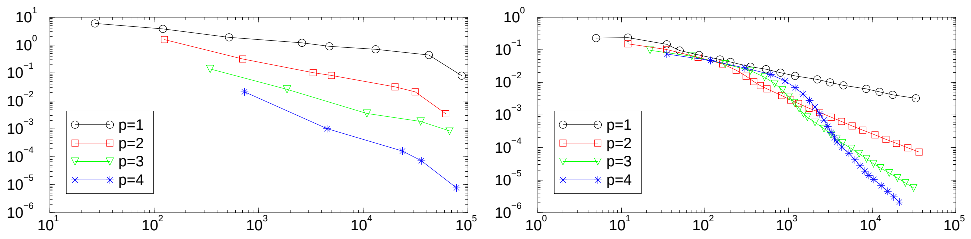

FIGURE 14. Comparison of error decay vs. DOFs for different polynomial degree p. Left: linear elasticity in 3D with a smooth solution; right: Poisson equation in 2D with a singular solution

and a finite element approximation u_h on the mesh $\mathcal{M}$, the local error indicators on single elements $E \in \mathcal{M}$ read like

$$\eta_E(u_h)^2 = C\, h_E^2 \|-\nabla\cdot A\nabla u_h + f(x,u_h,\nabla u_h)\|^2_{L^2(E)} + C\, h_E \|[A\nabla u_h]\|^2_{L^2(\partial E\cap\Omega)},$$

where h_E is the diameter of element E and $[\cdot]$ denotes the difference of (discontinuous) values on both sides of an element's boundary. The error estimate is then

$$\|\nabla u - \nabla u_h\|_{L^2(\Omega)} \le \eta(u_h) = \Big(\sum_{E\in\mathcal{M}} \eta_E(u_h)^2\Big)^{1/2}.$$

Higher order discretization. For most applications, the solution exhibits enough regularity in order to effectively apply a higher order finite element discretization. 'Effectively' means here, that the additional effort (more degrees of freedom at each mesh element, more dense matrices) for the higher order discretization results in a reasonable gain of precision or overall computational cost, by drastically reducing the total number of degrees of freedom needed for a given error tolerance, for example. Additionally, mixed methods may *need* higher order elements for a stable discretization of the problem, like the Taylor-Hood element in a mixed finite element method for the incompressible Navier-Stokes equations. Here, the discrete velocity and pressure spaces are given by globally continuous functions which are piecewise polynomials of degree p respectively $p-1$ for $p \ge 2$.

In Figure 14 we demonstrate the benefit from using a higher order discretization. The figure shows the error decay vs. number of degrees of freedom (DOFs) for polynomial degrees $p = 1, \dots, 4$ for a smooth solution of a linear elasticity problem and a solution of Poisson's equation with a point singularity. Especially for the smooth solution the higher order discretization is highly superior. The error on the first grid for quartic elements with less than 800 DOFs is already smaller than for the linears on the finest grid with nearly 100 000 DOFs. Even though the solution to Poisson's equation exhibits a point singularity, the higher order discretization pays off in the second example shown on the right, especially on finer meshes.

Section 6 presents applications from fluid mechanics and phase transitions, where different finite element spaces of various order are needed at the same time on locally adapted meshes.

4.1. **Design principles for the implementation of finite element methods.**

We continue by describing some implementational aspects of finite element methods on adaptive hierarchical meshes providing higher order ansatz functions. The ideas are inspired by the abstract concepts of finite elements. More details are presented in [30, 31] and [32]. The basic iteration of an adaptive finite element code for a stationary problem is

- Assemble and solve the discrete system;
- Calculate the error estimate;
- Adapt the mesh, when needed.

For time dependent problems, such an iteration is used in each time step, and the step size of a time discretization may be subject to adaptivity, too.

The core part of every finite element program is the problem dependent assembly and solution of the discretized problem. In the adaptive iteration, the assemblage and solution of a discrete system is necessary after each mesh change. Additionally, this step is usually the most time consuming part of that iteration.

A general finite element toolbox must provide flexibility in problems and finite element spaces while on the other hand this core part can be performed efficiently. Implementations based on hierarchical data structures, e.g. [2, 4, 5, 32], provide the basic ingredients needed by the most efficient tools for the solution of (non-) linear discrete problems, which are available now, namely multilevel preconditioners and multigrid solvers. Last but not least, such a hierarchy is directly used by the multiresolution visualization routines.

4.2. **Abstract data structures.** We start from the abstract concept of a finite element space as a triple consisting of a collection of mesh elements, a set of basis functions on an element, and the connection of local and global basis functions giving global *degrees of freedom* of a finite element function. This directly leads to the definition of three main groups of data structures:

- Data structures for geometric information storing the underlying mesh together with element coordinates, boundary type and geometry, etc.;
- Data structures for finite element information providing values of local basis functions and their derivatives;
- Data structures for algebraic information linking geometric data and finite element data.

Using these structures, a finite element toolbox provides the whole abstract framework like finite element spaces and adaptive strategies, together with hierarchical meshes, routines for mesh adaptation, and the complete administration of finite element spaces and the corresponding degrees of freedom during mesh modifications. The underlying data structures allow a flexible handling of such information.

The hierarchical mesh. The conforming simplicial mesh is generated by refinement of a given initial triangulation. Refined parts of the mesh can be de–refined (coarsened). The refinement and coarsening routines construct a sequence of nested meshes with a hierarchical structure. For the applications shown in Section 6 the recursive refinement by bisection is used (the "newest vertex" bisection in 2D, and Bänsch's [3] and Kossaczký's [21] algorithm in 3D). This makes sure that shape regularity of the triangulations is conserved. Figure 15 shows the *atomic refinement operations* in two and three dimensions. All elements meeting the common refinement edge are bisected at the same time; no other elements are involved in this atomic operation. Local coarsening is the inverse operation of a previous local refinement.

The bisectioning refinement of elements leads naturally to nested meshes with the hierarchical structure of *binary trees*, one tree for every element of the initial

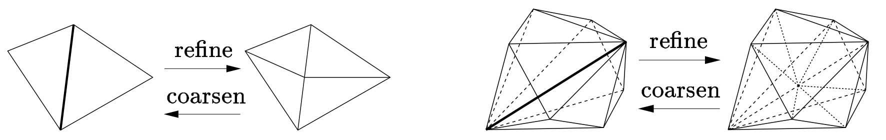

FIGURE 15. Atomic refinement and coarsening in 2D and 3D.

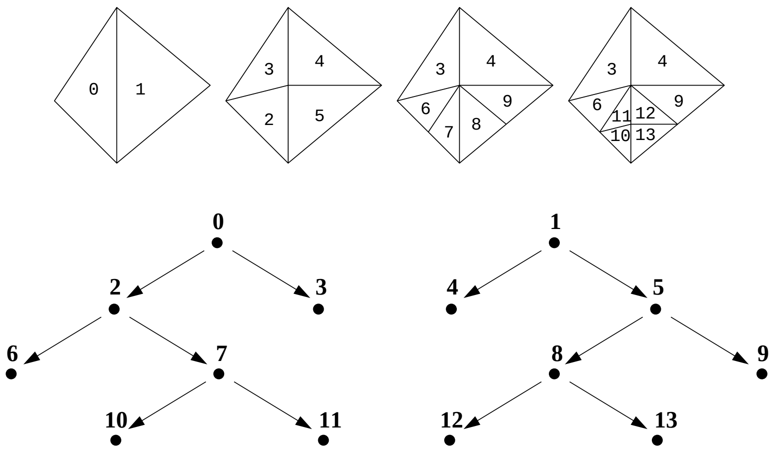

FIGURE 16. Some 2D mesh refinement and the corresponding binary trees.

triangulation. Every interior node of that tree has two pointers to the two children; the leaf elements are part of the actual triangulation, which is used to define the finite element space. The whole triangulation is a list of given macro elements together with the associated binary trees, see Figure 16 for a simple 2D example. These binary trees correspond to the h-hierarchy part of the abstract trees considered in Section 1.

The hierarchical structure allows the generation of most geometrical information (like coordinates of vertices or element adjacencies) by the hierarchy and data from the macro triagulation, which reduces the amount of data to be stored. Furthermore, the hierarchical mesh structure directly leads to multilevel information which can be used by multilevel preconditioners and solvers. Access to mesh elements is available solely via routines which traverse the hierarchical trees. For example, such tree traversal routines implement the interface between numerical data and visualization, compare Section 5.

Finite elements. The values of a finite element function or the values of its derivatives are uniquely defined by the values of its DOFs and the values of the basis functions or the derivatives of the basis functions connected with these DOFs. We follow the concept of finite elements which are given on a single element E in local coordinates: Finite element functions on an element E are defined by a finite dimensional function space $\bar{\mathbb{P}}$ on a reference element $\bar{E}$ and the (one to one) mapping $\lambda^E : \bar{E} \to E$ from the reference element $\bar{E}$ to the element E. Also, derivatives are given by the derivatives of basis functions on $\bar{\mathbb{P}}$ and derivatives of λ^E. The matrices S_i^q defined in Section 3.1 are sample-matrices of a special basis of $\bar{\mathbb{P}} = \mathbb{P}(\bar{E}, q)$ on the reference element $\bar{E} = E_0$.

Degrees of freedom. Finite element data is connected with geometric information of a triangulation by the degrees of freedom. For piecewise polynomial finite element functions of order p with a Lagrange basis, the values in all Lagrange nodes $\bigcup_{E\in\mathcal{M}}\mathcal{L}(E,p)$ build the associated degrees of freedom (for $p = 1$ these are the values in all vertices of the triangulation). For general applications, it is necessary to handle several different sets of degrees of freedom on the same triangulation. For example, in mixed finite element methods for the Navier-Stokes problem, different polynomial degrees are used for discrete velocity and pressure functions.

The DOF administration tool gives access from an element E to the global degree of freedom corresponding to a local basis function. During mesh refinement and coarsening, DOFs are created or removed. The DOF administration takes care of the varying number of DOFs and corresponding length adjustments for coefficient vectors or matrices associated to the finite element spaces.

4.3. **Application problem solving in 2D and 3D.** Both geometric and finite element information strongly depend on the space dimension. Thus, mesh modification algorithms and basis functions are implemented for 2D and 3D separately and are provided by the toolbox. Everything besides that can be formulated in such a way that the dimension only enters as a parameter (like size of local coordinate vectors, e.g.). For usual finite element applications this results in a dimension independent programming, where all dimension dependent parts are hidden in a library. Hence, program development can be done in 2D, where execution is much faster and debugging is much easier. With no (or maybe few) additional changes, the program will then also work in 3D. This approach leads to a tremendous reduction of program development time for 3D problems.

The design principles and data structures described above are realized in the finite element toolbox ALBERT [30, 31, 32].

5. The interface between visualization and numerics

It is very important that visualization supports the numerical data structure in order to benefit from the implemented grid handling. From the visualization point of view there are mainly two possibilities of data access. Most frequently used visualization software works on prescribed data formats. The numerical data structures have to be converted into such a format. In contrast to this, in a procedural data access the numerical data structures are addressed directly by functions. The visualization tools directly work on the data structures used in the numerical applications. One has to provide some access procedures and give a description of the element types. In what follows we will briefly describe the idea of a procedural data access between visualization and numerics.

In [29] a visualization interface for arbitrary meshes with general data functions on them has been proposed. A mesh is defined as a procedurally linked list of non intersecting elements. The access to data is done by user supplied procedures addressing the user data structures and returning the required data temporarily in a prescribed element structure. This structure especially contains a polygonal boundary representation, the coordinate vectors for the nodes and function data on them. Combining such information with local coordinates for the elements, a large class of element types can be handled. (Here we restrict ourselves to the basic concept. The true data structures are slightly more general, especially concerning

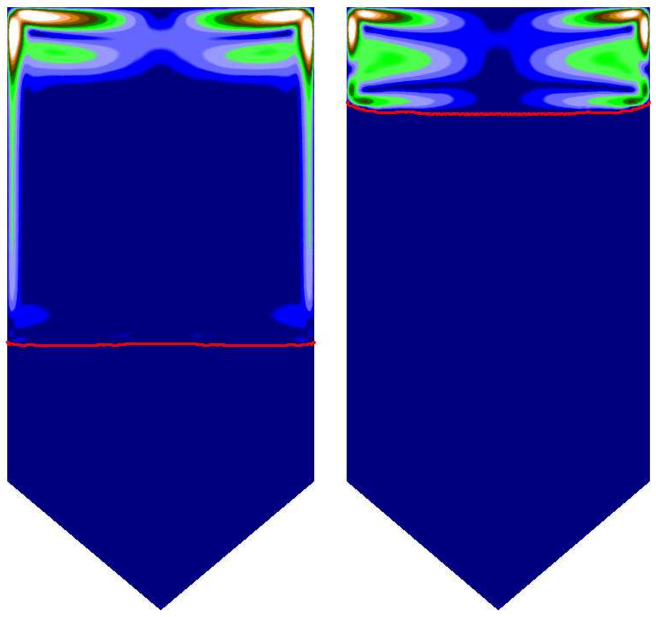

FIGURE 17. Semiconductor crystal growth by the vertical Bridgman method; solid–liquid interface (red) and magnitude of convection in the melt at two different times.

the interface for function data (cf. [29]).) As the rendering procedures only take into account these representations of an element, the approach applies to a wide range of commonly used unstructured grids. In [25] and [26] various applications on different mesh types in two and three dimensions are shown.

The access routines to the elements in the hierarchical structure are implemented by directly using the traversal routines of the numerical code (cf. Section 4.2). For a detailed description of the access procedures we refer to [24, 25]. Through the procedural access a visualization method is supplied with all necessary information to locally evaluate and graphically represent grid geometry and data. This is sufficient to run merely all visualization algorithms, e. g. isosurface rendering and slicing combined with a color shading on the generated slices.

6. APPLICATIONS

In order to demonstrate the flexibility of adaptive higher order finite element methods and the benefits of polynomial and multiresolution visualization, we present two applications in CFD. The first application is a combination of phase transition and fluid flow. It models the Bridgman growth of a semiconductor crystal with thermal convection in the melt. An ampoule containing the molten material is slowly moved inside an oven to a colder zone such that the material solidifies, at best resulting in a single crystal. For the simulation, the classical Stefan problem is coupled to the Navier–Stokes equations using a fictitious domain approach: The Stefan equation contains an additional convection term including the flow velocity in the melt, while the flow is driven by a force resulting from temperature and gravity (in the Boussinesq approximation). Figure 17 shows the solid–liquid interface

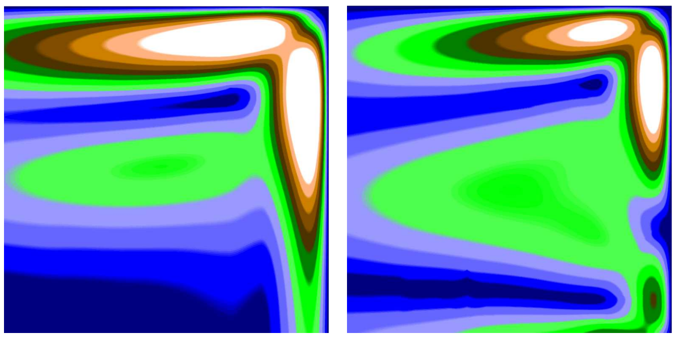

FIGURE 18. Crystal growth; zoom to the upper right corners of the ampoules (cf. Figure 17).

(in red) and the flow velocity from a 2D simulation at two different time steps [6, 7]. Figure 18 contains a zoom of both time steps. The flow pattern is given by two narrow convection rolls, one at the top and another one moving upwards above the interface. Near the end of the growth process they meet, as is shown in the pictures on the right hand side. The strongest overall convection is observed around this time. The numerical approximation for the Navier–Stokes equations uses a $\mathbb{P}_3$–$\mathbb{P}_2$ Taylor–Hood element; linear finite elements are used for the Stefan problem. The piecewise smoothness of the given data is very well visible by using p–textures.

The second application is the investigation of air flow in a flue pipe, especially the initiation of acoustic waves by the air flow [1]. The flow is modeled by the incompressible Navier–Stokes equations. Figure 19 shows the air velocity in a part of a 2D model geometry for the flue pipe around the labium at two different times. The geometry is taken from [35]. Air enters from the thin inlet channel on the left and is split at the labium. The numerical approximation uses the classical $\mathbb{P}_2$–$\mathbb{P}_1$ Taylor–Hood element. The piecewise smooth nature of given data is again perfectly resolved.

The results for the h–p–multiresolution are shown in Figure 20 for threshold value $\varepsilon = 0.35$. The algorithm detects the region around the labium to contain most visual information and therefore uses maximum polynomial degree (=2) for visualization. Other regions are resolved with lower polynomial degree, some elements even are not resolved to the leaf-level. The consistency of the algorithm and the detection of fine details is perfectly demonstrated.

7. Conclusion

We have introduced a h–p–multiresolution visualization framework for the post processing of fairly general data from h–p–adaptive finite element computations. It extends recent work for multi-linear finite element data [25, 26, 28] in a very natural way. By providing polynomial error indicators and extending the saturation procedure we obtain adaptive projections on locally varying grid levels and polynomial degrees. Nevertheless continuity of the extracted projection is ensured. We further demonstrated the applicability by choosing polynomial textures as one visualization method for polynomial data and applying this to actual finite element simulation results. Compared to piecewise linear extraction methods the h–p–multiresolution visualization produces much better results. The required computational time on

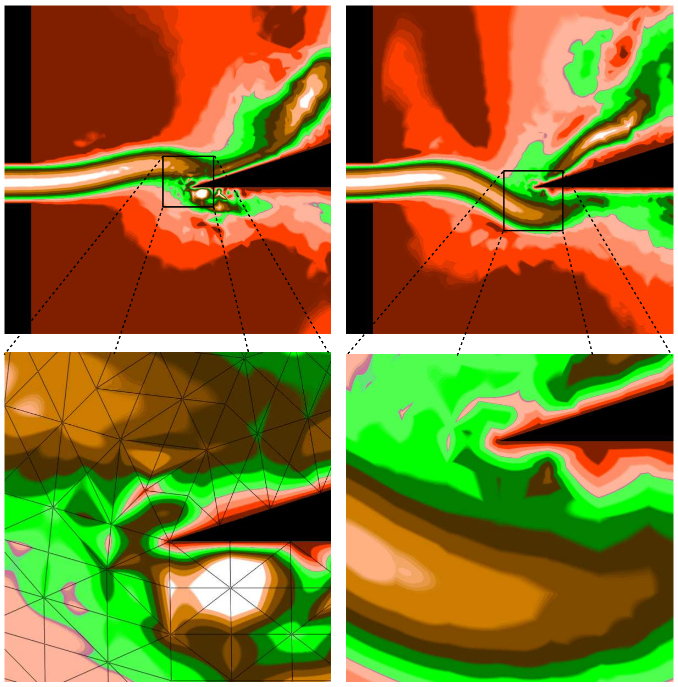

FIGURE 19. Air flow in a flue pipe; the upper pictures show velocities near the labium at two different times, the lower pictures contain zooms to the labium tip region. The lower left picture additionally indicates the grid used for visualization.

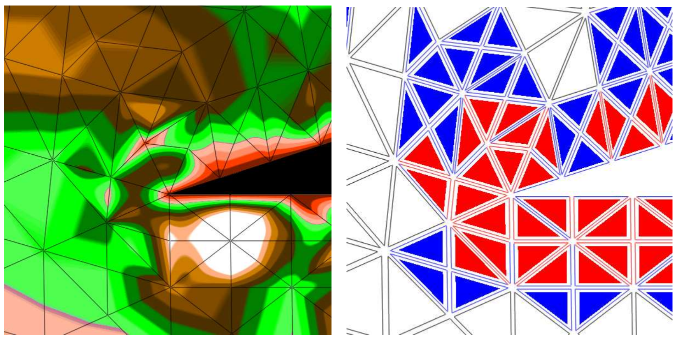

FIGURE 20. Adaptive visualization of the left timestep of Figure 19 with $\varepsilon = 0.35$. The right picture indicates the polynomial degrees used for the elements and edges. Red represents polynomial degree 2, blue polynomial degree 1. White elements are non-leaf elements.

each cell is optimal if we neglect the required computations in texture hardware. In fact it depends linearly on the number of polynomial basis functions involved. Thus the new method is outperforming any piecewise linear subsampling approach.

References

[1] A. Bamberger, E. Bänsch, and K. G. Siebert. Experimental and numerical investigation of edge tones. Preprint WIAS Berlin, 2001.

[2] R. E. Bank. PLTMG: a software package for solving elliptic partial differential equations user's guide 8.0. Software - Environments - Tools. 5. Philadelphia, PA: SIAM. xii, 1998.

[3] E. Bänsch. Local mesh refinement in 2 and 3 dimensions. *IMPACT Comput. Sci. Eng.*, 3:181–191, 1991.

[4] P. Bastian, K. Birken, K. Johannsen, S. Lang, V. Reichenberger, C. Wieners, G. Wittum, and C. Wrobel. Parallel solution of partial differential equations with adaptive multigrid methods on unstructured grids. In E. Krause and et al., editors, *High performance computing in science and engineering '99*, pages 496–508. Berlin: Springer, 2000. Transactions of the High Performance Computing Center Stuttgart (HLRS). 2nd workshop, Stuttgart, Germany, October 4-6, 1999.

[5] R. Beck, B. Erdmann, and R. Roitzsch. An object-oriented adaptive finite element code: Design issues and applications in hyperthermia treatment planning. In E. Arge and et al., editors, *Modern software tools for scientific computing*, pages 105–124. Boston: Birkhaeuser, 1997. International workshop, Oslo, Norway, September 16–18, 1996.

[6] St. Boschert, A. Schmidt, and K. G. Siebert. Numerical simulation of crystal growth by the vertical Bridgman method. In J.S. Szmyd and K. Suzuki, editors, *Modelling of Transport Phenomena in Crystal Growth*, Development in Heat Transfer Series. WIT Press, 2000.

[7] St. Boschert, A. Schmidt, K. G. Siebert, E. Bänsch, K.W. Benz, G. Dziuk, and T. Kaiser. Simulation of industrial crystal growth by the vertical Bridgman method. Report 00-11 ZeTeM Bremen and Preprint 14/2000 Freiburg. To appear.

[8] A. Certain, J. Popović, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *SIGGRAPH 96 Conference Proceedings*, pages 91–98, 1996.

[9] P.G. Ciarlet and J.L. Lions (eds.). *Handbook of numerical analysis. Vol. V: Techniques of scientific computing.* Elsevier, 1997.

[10] P. Cignoni, L. De Floriani, C. Montoni, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *1994 Symposium on Volume Visualization*, pages 19–26, 1994.

[11] P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13(5):199–217, 1997. ISSN 0178-2789.

[12] L. De Floriani, P. Marzano, and E. Puppo. Multiresolution models for topographic surface description. *The Visual Computer*, 12(7):317–345, 1996.

[13] N. Faust, L.F. Hodges, D. Koller, P. Lindstrom, W. Ribarsky, and G.A. Turner. Real-time, continuous level of detail rendering of hight fields. *Computer Graphics & Applications*, pages 109–117, 1996.

[14] T. Gerstner, M. Rumpf, and U. Weikard. Error indicators for multilevel visualization and computing on nested grids. *Computers & Graphics*, 24(3):363–373, 2000.

[15] GRAPE. GRAphics Programming Environment: Reference Manual. http://www.mathematik.uni-freiburg.de/Grape/DOC/HTML/manual.html, Institut für Angewandte Mathematik, Universität Freiburg, 1996.

[16] M. H. Gross and R. G. Staadt. Fast multiresolution surface meshing. In *Proceedings of the Visualization*, pages 135–142, 1995.

[17] R. Grosso, Ch. Luerig, and Th. Ertl. The multilevel finite element method for adaptive mesh optimization and visualization of volume data. In *Proceedings Visualization*. IEEE, 1997.

[18] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11:197–214, 1994.

[19] M. Hopf and T. Ertl. Accelerating 3D convolution using graphics hardware. In D. Ebert, M. Gross, and B. Hamann, editors, *IEEE Visualization '99*, pages 471–474, San Francisco, 1999. IEEE.

[20] H. Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings*, pages 99–108, 1996.

[21] I. Kossaczký. A recursive approach to local mesh refinement in two and three dimensions. *J. Comput. Appl. Math.*, 55:275–288, 1994.

[22] D. Laur and P. Hanrahan. Hierarchical splatting: a progressive refinement algorithm for volume rendering. *Computer Graphics*, 25(4):285–288, July 1991.

[23] L. Lippert and M. H. Gross. Fast wavelet based volume rendering by accumulation of transparent texture maps. *Computer Graphics Forum*, 14(3):431–444, 1995. Figures: http://www.inf.ethz.ch/personal/lippert/PAPERS/228fig6-11.rgb.gz.

[24] R. Neubauer, M. Ohlberger, M. Rumpf, and R. Schwörer. Efficient visualization of large-scale data on hierarchical meshes. Preprint, IAM, Universität Freiburg, 1997.

[25] M. Ohlberger and M. Rumpf. Hierarchical and Adaptive Visualization on Nested Grids. *Computing*, 59:365–385, 1997.

[26] M. Ohlberger and M. Rumpf. Adaptive Projection Operators in Multiresolution Scientific Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):74–94, 1999.

[27] L. Piegl and W. Tiller. *The NURBS Book.* Monographs in visual communication. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., second edition, 1997.

[28] M. Rumpf. Recent numerical methods – a challenge for data analysis and visualization. *Future General Computer Systems*, 15:43–58, 1999.

[29] M. Rumpf, A. Schmidt, and K.G. Siebert. Functions defining arbitrary meshes, a flexible interface between numerical data and visualization routines. *Computer Graphics Forum*, 15:129–141, 1996.

[30] A. Schmidt and K. G. Siebert. ALBERT: An adaptive hierarchical finite element toolbox. Preprint 06/2000 Freiburg, 2000. Documentation.

[31] A. Schmidt and K. G. Siebert. ALBERT – Software for scientific computations and applications. *Acta Math. Univ. Comenianae.*, 70:105–122, 2001.

[32] A. Schmidt and K.G. Siebert. Concepts of the finite element toolbox ALBERT. Preprint 17/98 Freiburg, 1998. To appear in Notes on Numerical Fluid Mechanics.

[33] W. J. Schroeder, J. A. Zarge, and W. A. Lorensen. Decimation of triangle meshes. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, 1992.

[34] R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill. Octree-based decimation of marching cubes surfaces. In *Proceedings Visualization.* IEEE, 1996.

[35] P.A. Skordos. Modeling flue pipes: subsonic flow, lattice Boltzmann, and parallel distributed computers. PhD thesis, Massachusetts Institute of Technology, 1995.

[36] R. Strzodka and M. Rumpf. Level set segmentation in graphics hardware. In *Proceedings ICIP 2001*, to appear.

[37] R. Strzodka and M. Rumpf. Nonlinear diffusion in graphics hardware. In *Proceedings VisSym 2001*, to appear.

[38] G. Turk. Re-tiling polygonal surfaces. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[39] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. *Computer Graphics (SIGGRAPH '98)*, 32(4):169–179, 1998.

[40] M. Wierse and M. Rumpf. GRAPE, Eine interaktive Umgebung für Visualisierung und Numerik. *Informatik, Forschung und Entwicklung, Springer-Verlag*, 7:145–151, 1992.

[41] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing volume data. In *Proceedings Visualization.* IEEE, 1997.

[42] G. W. Zumbusch. Visualizing functions of the h-p-version of finite elements. Technical Report TR-94-05, Konrad-Zuse-Zentrum, Berlin, Germany, 1994.

Bernard Haasdonk, Institut für Informatik, Universität Freiburg, Georges-Köhler-Allee Geb. 52, D-79110 Freiburg
E-mail address: haasdonk@informatik.uni-freiburg.de

Mario Ohlberger, Institut für Angewandte Mathematik, Universität Freiburg, Hermann-Herder-Str. 10, D-79104 Freiburg
E-mail address: mario@mathematik.uni-freiburg.de

Martin Rumpf, Fachbereich Mathematik, Universität Duisburg, Lotharstrasse 65, D-47048 Duisburg
E-mail address: rumpf@math.uni-duisburg.de

Alfred Schmidt, Zentrum für Technomathematik, FB 3, Universität Bremen, Postfach 33 04 40, D-28334 Bremen
E-mail address: schmidt@math.uni-bremen.de

Kunibert G. Siebert, Institut für Angewandte Mathematik, Universität Freiburg, Hermann-Herder-Str. 10, D-79104 Freiburg
E-mail address: Kunibert.Siebert@mathematik.uni-freiburg.de

Berichte aus der Technomathematik ISSN 1435-7968

`http://www.math.uni-bremen.de/zetem/berichte.html`

— Vertrieb durch den Autor —

Reports **Stand: 10. Oktober 2001**

98–01. Peter Benner, Heike Faßbender:
An Implicitly Restarted Symplectic Lanczos Method for the Symplectic Eigenvalue Problem, Juli 1998.

98–02. Heike Faßbender:
Sliding Window Schemes for Discrete Least-Squares Approximation by Trigonometric Polynomials, Juli 1998.

98–03. Peter Benner, Maribel Castillo, Enrique S. Quintana-Ortí:
Parallel Partial Stabilizing Algorithms for Large Linear Control Systems, Juli 1998.

98–04. Peter Benner:
Computational Methods for Linear–Quadratic Optimization, August 1998.

98–05. Peter Benner, Ralph Byers, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Solving Algebraic Riccati Equations on Parallel Computers Using Newton's Method with Exact Line Search, August 1998.

98–06. Lars Grüne, Fabian Wirth:
On the rate of convergence of infinite horizon discounted optimal value functions, November 1998.

98–07. Peter Benner, Volker Mehrmann, Hongguo Xu:
A Note on the Numerical Solution of Complex Hamiltonian and Skew-Hamiltonian Eigenvalue Problems, November 1998.

98–08. Eberhard Bänsch, Burkhard Höhn:
Numerical simulation of a silicon floating zone with a free capillary surface, Dezember 1998.

99–01. Heike Faßbender:
The Parameterized SR Algorithm for Symplectic (Butterfly) Matrices, Februar 1999.

99–02. Heike Faßbender:
Error Analysis of the symplectic Lanczos Method for the symplectic Eigenvalue Problem, März 1999.

99–03. Eberhard Bänsch, Alfred Schmidt:
Simulation of dendritic crystal growth with thermal convection, März 1999.

99–04. Eberhard Bänsch:
Finite element discretization of the Navier-Stokes equations with a free capillary surface, März 1999.

99–05. Peter Benner:
Mathematik in der Berufspraxis, Juli 1999.

99–06. Andrew D.B. Paice, Fabian R. Wirth:
Robustness of nonlinear systems and their domains of attraction, August 1999.

99–07. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Balanced Truncation Model Reduction of Large-Scale Dense Systems on Parallel Computers, September 1999.

99–08. Ronald Stöver:
Collocation methods for solving linear differential-algebraic boundary value problems, September 1999.

99–09. Huseyin Akcay:
Modelling with Orthonormal Basis Functions, September 1999.

99–10. Heike Faßbender, D. Steven Mackey, Niloufer Mackey:
Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems, Oktober 1999.

99–11. Peter Benner, Vincente Hernández, Antonio Pastor:
On the Kleinman Iteration for Nonstabilizable System, Oktober 1999.

99–12. Peter Benner, Heike Faßbender:
A Hybrid Method for the Numerical Solution of Discrete-Time Algebraic Riccati Equations, November 1999.

99–13. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Numerical Solution of Schur Stable Linear Matrix Equations on Multicomputers, November 1999.

99–14. Eberhard Bänsch, Karol Mikula:
Adaptivity in 3D Image Processing, Dezember 1999.

00–01. Peter Benner, Volker Mehrmann, Hongguo Xu:
Perturbation Analysis for the Eigenvalue Problem of a Formal Product of Matrices, Januar 2000.

00–02. Ziping Huang:
Finite Element Method for Mixed Problems with Penalty, Januar 2000.

00–03. Gianfrancesco Martinico:
Recursive mesh refinement in 3D, Februar 2000.

00–04. Eberhard Bänsch, Christoph Egbers, Oliver Meincke, Nicoleta Scurtu:
Taylor-Couette System with Asymmetric Boundary Conditions, Februar 2000.

00–05. Peter Benner:
Symplectic Balancing of Hamiltonian Matrices, Februar 2000.

00–06. Fabio Camilli, Lars Grüne, Fabian Wirth:
A regularization of Zubov's equation for robust domains of attraction, März 2000.

00–07. Michael Wolff, Eberhard Bänsch, Michael Böhm, Dominic Davis:
Modellierung der Abkühlung von Stahlbrammen, März 2000.

00–08. Stephan Dahlke, Peter Maaß, Gerd Teschke:
Interpolating Scaling Functions with Duals, April 2000.

00–09. Jochen Behrens, Fabian Wirth:
A globalization procedure for locally stabilizing controllers, Mai 2000.

00–10. Peter Maaß, Gerd Teschke, Werner Willmann, Günter Wollmann:
Detection and Classification of Material Attributes – A Practical Application of Wavelet Analysis, Mai 2000.

00–11. Stefan Boschert, Alfred Schmidt, Kunibert G. Siebert, Eberhard Bänsch, Klaus-Werner Benz, Gerhard Dziuk, Thomas Kaiser:
Simulation of Industrial Crystal Growth by the Vertical Bridgman Method, Mai 2000.

00–12. Volker Lehmann, Gerd Teschke:
Wavelet Based Methods for Improved Wind Profiler Signal Processing, Mai 2000.

00–13. Stephan Dahlke, Peter Maass:
A Note on Interpolating Scaling Functions, August 2000.

00–14. Ronny Ramlau, Rolf Clackdoyle, Frédéric Noo, Girish Bal:
Accurate Attenuation Correction in SPECT Imaging using Optimization of Bilinear Functions and Assuming an Unknown Spatially-Varying Attenuation Distribution, September 2000.

00–15. Peter Kunkel, Ronald Stöver:
Symmetric collocation methods for linear differential-algebraic boundary value problems, September 2000.

00–16. Fabian Wirth:
The generalized spectral radius and extremal norms, Oktober 2000.

00–17. Frank Stenger, Ahmad Reza Naghsh-Nilchi, Jenny Niebsch, Ronny Ramlau:
A unified approach to the approximate solution of PDE, November 2000.

00–18. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Parallel algorithms for model reduction of discrete–time systems, Dezember 2000.

00–19. Ronny Ramlau:
A steepest descent algorithm for the global minimization of Tikhonov–Phillips functional, Dezember 2000.

01–01. Efficient methods in hyperthermia treatment planning:
Torsten Köhler, Peter Maass, Peter Wust, Martin Seebass, Januar 2001.

01–02. Parallel Algorithms for LQ Optimal Control of Discrete-Time Periodic Linear Systems:
Peter Benner, Ralph Byers, Rafael Mayo, Enrique S. Quintana-Ortí, Vicente Hernández, Februar 2001.

01–03. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Efficient Numerical Algorithms for Balanced Stochastic Truncation, März 2001.

01–04. Peter Benner, Maribel Castillo, Enrique S. Quintana-Ortí:
Partial Stabilization of Large-Scale Discrete-Time Linear Control Systems, März 2001.

01–05. Stephan Dahlke:
Besov Regularity for Edge Singularities in Polyhedral Domains, Mai 2001.

01–06. Fabian Wirth:
A linearization principle for robustness with respect to time-varying perturbations, Mai 2001.