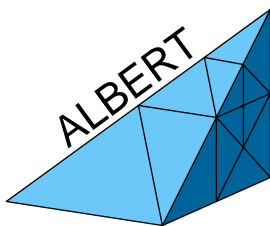


## ALBERT— SOFTWARE FOR SCIENTIFIC COMPUTATIONS AND APPLICATIONS

A. SCHMIDT AND K. G. SIEBERT

ABSTRACT. Adaptive finite element methods are a modern, widely used tool which make realistic computations feasible, even in three space dimensions. We describe the basic ideas and ingredients of adaptive FEM and the implementation of our toolbox ALBERT. The design of ALBERT is based on the natural hierarchy of locally refined meshes and an abstract concept of general finite element spaces. As a result, dimension independent programming of applications is possible. Numerical results from applications in two and three space dimensions demonstrate the flexibility of ALBERT.

### 1. INTRODUCTION



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, by using 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, specially in 3d.

Finite element methods calculate approximations to the true solution in some finite dimensional function space. This space is built up from **local function spaces**, usually polynomials of low order, on elements of a partitioning of the domain (the **mesh**). An adaptive  $h$ -method adjusts 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 [2, 3, 17].

---

Received January 10, 2001.

2000 *Mathematics Subject Classification*. Primary 65N30, 65N50, 65Y15.

*Key words and phrases*. Adaptive finite element methods, scientific software, software design.

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 on some reference element and the mapping from this reference element to the mesh elements.

The mesh adaptation is based on information extracted from **a posteriori error estimators**. These estimators are **computable** estimates for the error between the true solution and the finite element approximation and they are built 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.

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 the 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. This holds for programs that solve the discrete problem on a fixed mesh as well as for adaptive methods that automatically adjust the underlying mesh to the actual problem and solution. 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. Data structures are needed which allow an easy and efficient implementation of the problem dependent parts and also allow to use adaptive methods, mesh modification algorithms, and solvers for linear and nonlinear discrete problems by calling library routines. On the one hand, large flexibility is needed in order to choose various kinds of finite element spaces, with higher order elements or combinations of different spaces for mixed methods or systems. On the other hand, the solution of the resulting discrete systems may profit enormously from a simple vector-oriented storage of coefficient vectors and matrices. This also allows the use of optimized solver and BLAS libraries. Additionally, implementations based on hierarchical data structures 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 [6, 9, 10, 29].

During the last years there has been a great progress in designing finite element software. Besides the already cited packages [6, 9, 10, 29], a list of freely available finite element codes can for instance be found at

[http://www.engr.usask.ca/~macphed/finite/fe\\_resources](http://www.engr.usask.ca/~macphed/finite/fe_resources).

In the next sections, we describe the design of our finite element toolbox ALBERT with adaptive hierarchical meshes providing higher order ansatz functions. ALBERT provides all the tools mentioned above for the efficient implementation and adaptive solution of general nonlinear problems in two and three space dimensions. The mesh adaptation is done by local refinement and coarsening of mesh elements, while the same local function space is used on all mesh elements. The design of the ALBERT data structures allows an implementation of problem dependent parts that does not depend on the actual chosen local function space and the dimension.

The paper is organized as follows: in Section 2 we recall the adaptive finite element method, Section 3 demonstrates the benefit of a higher order discretization and in Section 4 we describe the basic data structures of ALBERT. We conclude with Section 5 presenting several applications from research projects in order to demonstrate the flexibility of ALBERT.

## 2. ADAPTIVE FINITE ELEMENTS

The basic ingredient of an adaptive finite element method is the a posteriori error estimator. While a general theory exists for these estimators in the case of linear and mildly nonlinear problems [8, 32], highly nonlinear problems usually still need a special treatment [15, 21, 27, 30].

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., a fairly incomplete list of references is [1, 4, 5, 11, 12, 20, 25, 31]. The a posteriori estimate is usually derived by using the differential equation to eliminate the true solution and replacing it by given data in an appropriate error representation; thus, the estimator involves given data of the equation, compare the example below.

Here, we just want to give an example for residual estimators, which are also used in the simulations shown later. For a nonlinear (quasi-linear) elliptic problem

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

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

$$\eta_S^2 = C_0 h_S^2 \| -\nabla \cdot A \nabla u_S + f(x, u_S, \nabla u_S) \|_{L^2(S)}^2 + C_1 h_S \| [A \nabla u_S \cdot \nu] \|_{L^2(\partial S \setminus \partial\Omega)}^2,$$

where  $C_0, C_1$  depend on interpolation constants and the differential operator,  $h_S$  is the diameter of element  $S$ ,  $\nu$  is the normal to  $\partial S$  and  $[\cdot]$  denotes the difference of (discontinuous) values on both sides of an element's boundary. The error estimate is then

$$\| \nabla(u - u_S) \|_{L^2(\Omega)} \leq \eta = \left( \sum_{S \in \mathcal{S}} \eta_S^2 \right)^{1/2}.$$

The aim of adaptive methods is the generation of a mesh 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 philosophy implemented in ALBERT is to change meshes successively by local refinement or coarsening using information produced by a posteriori error estimators.

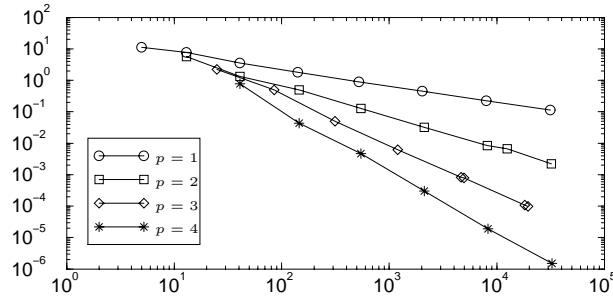
Several adaptive strategies are proposed in the literature, that give criteria which mesh elements should be marked for refinement (see [4, 19, 20, 24] for examples). Most strategies are based on the idea of an equidistribution of the local error to all mesh elements. Babuška and Rheinboldt [4] motivate that for stationary problems a mesh is almost optimal when the local errors are approximately equal for all elements. So, elements where the error indicator is large will be marked for refinement, while elements with a small indicator are left unchanged or are marked for coarsening. In time dependent problems, the mesh is adapted to the solution in every time step using a posteriori information like in the stationary case. As a first trial mesh for the new time step we use the adaptive mesh from the previous time step. Usually, only few iterations of the adaptive procedure are then needed for the adaptation of the mesh for the new time step. This may be accompanied by an adaptive control of time step sizes.

### 3. HIGHER ORDER DISCRETIZATION

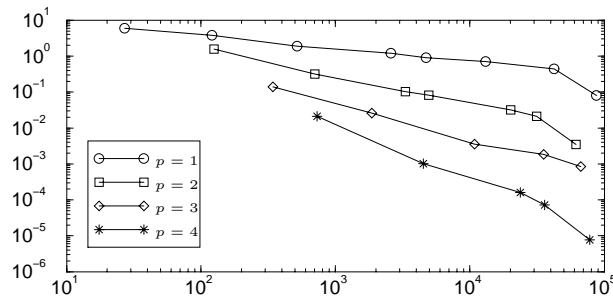
There exist some applications where only piecewise linear ansatz functions can be used for discretization, because they provide special monotonicity and maximum properties. Examples include nonlinear models with obstacle problems and nonlinear monotone operators, e.g., where even the exact solution does not show higher regularity, at least in some parts of the domain [15, 22, 26].

However, 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 build from globally continuous functions which are piecewise polynomials of degree  $p$  respectively  $p - 1$  for  $p \geq 2$ .

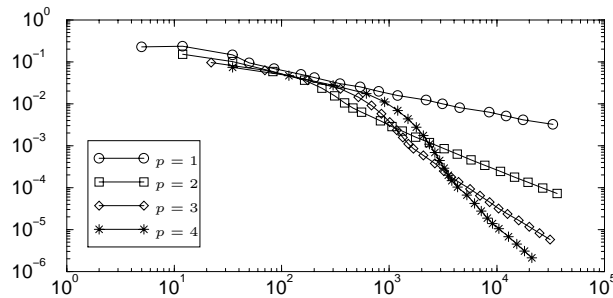
In Figures 1–3 we demonstrate the benefit from using a higher order discretization in an adaptive finite element method. The figures show the error decay vs. number of degrees of freedoms (DOFs) for polynomial degrees  $p = 1, \dots, 4$  for problems from linear elasticity (Figures 1 respectively 2) with smooth solutions and Figure 3 refers to a solution of Poisson’s equation with a corner singularity. Especially for the smooth solution the higher order discretization is much superior. For the 2d example the error on the first grid for quartic elements with  $\approx 50$  DOFs



**Figure 1.** Comparison of error decay vs. DOFs for different polynomial degree for linear elasticity in 2d with a smooth solution.



**Figure 2.** Comparison of error decay vs. DOFs for different polynomial degree for linear elasticity in 3d with a smooth solution.



**Figure 3.** Comparison of error decay vs. DOFs for different polynomial degree for Poisson equation in 2d with a solution exhibiting a point singularity.

is nearly the same as for the linears on the finest grid with  $\approx 50\,000$  DOFs. The same holds for the 3d example where this effect is even stronger. Even though the solution to Poisson's equation exhibits a point singularity, the higher order discretization pays off in the last example (Figure 3). The curves corresponding to the higher order discretizations are always below the curve of the piecewise linear discretization. Here, the piecewise quadratic perform the best on coarser grid, but are then passed by the piecewise cubic and quartic discretizations on finer meshes.

Note, that without adaptivity, the higher order discretization would perform worse than the piecewise linear discretization in this example.

#### 4. ABSTRACT DATA STRUCTURES FOR A FINITE ELEMENT PACKAGE

Starting point for the design of ALBERT data structures is the abstract concept of a finite element space defined (similar to the definition of a single finite element by Ciarlet [16]) as a triple consisting of

- a collection of **mesh elements**;
- a set of local **basis functions** on a single element, usually a restriction of global basis functions to a single element;
- a 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 data structures, the finite element toolbox ALBERT 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 (DOFs) during mesh modifications. The underlying data structures allow a flexible handling of such information. Furthermore, tools for numerical quadrature, matrix and load vector assembly as well as solvers for linear and nonlinear problems, like conjugate gradient methods and Newton's method, are available.

A specific problem can be implemented and solved by providing just some problem dependent routines for evaluation of the (linearized) differential operator, data, nonlinear solver, and (local) error estimators, using all the above mentioned tools from the library.

Both geometric and finite element information strongly depend on the space dimension. Thus, mesh modification algorithms and basis functions are implemented for two (2d) and three (3d) dimensions 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. This allows a dimension independent programming of applications to the greatest possible extent.

#### 4.1. The Hierarchical Mesh

The underlying mesh is a conforming triangulation of the computational domain into simplices, i.e. triangles (2d) or tetrahedra (3d). The simplicial mesh is generated by refinement of a given initial triangulation. Refined parts of the mesh can be de-refined, but elements of the initial triangulation (**macro elements**) must not be coarsened. The refinement and coarsening routines construct a sequence of nested meshes with a hierarchical structure. In ALBERT, the recursive refinement by bisection is implemented. For every element (triangle or tetrahedron) one of its edges is marked as the refinement edge, and during refinement the element is bisected into two elements by cutting this edge at its midpoint, see Figure 4 where the refinement edge is indicated by a bold line.



Figure 4. Bisection of a triangle or tetrahedron.

We use algorithms where the choice of refinement edges on the initial triangulation prescribes the refinement edges for all simplices that are created during mesh refinement (the “newest vertex” bisection in 2d, and Bänsch’s [7] and Kossaczky’s [23] algorithms in 3d). This makes sure that shape regularity of the triangulations is conserved. The use of the recursive refinement procedure results in an **atomic refinement operation** which is shown in Figure 5 for 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.

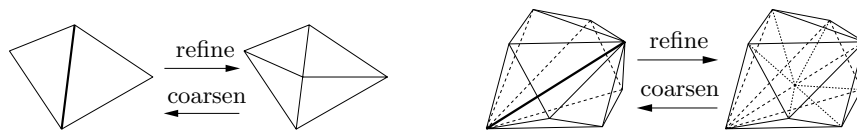
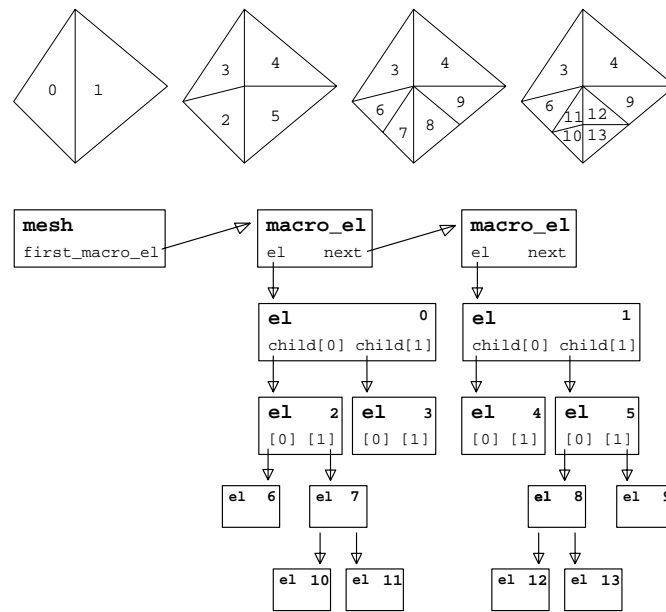


Figure 5. Atomic refinement and coarsening in 2d and 3d.

During refinement, new degrees of freedom are created. DOFs are stored (and accessed) element-wise (compare Section 4.3) and a single DOF is shared by all elements which belong to the support of the corresponding finite element basis function (compare Section 4.2). The mesh refinement routines must create a new DOF only once and give access to this DOF from all elements sharing it. Similarly, DOFs are handled during coarsening. This is done in cooperation with the DOF administration tool, see below.

The bisectioning refinement of elements naturally leads to nested meshes with the hierarchical structure of binary trees, one tree for every element of the initial 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 6 for a simple 2d example.



**Figure 6.** Some 2d mesh refinement and the corresponding binary trees.

The hierarchical structure allows the generation of most information by the hierarchy, which reduces the amount of data to be stored. Some information is stored on the (leaf) elements explicitly, other information is located at the macro elements and is transferred to the leaf elements while traversing through the binary tree. Element information about vertex coordinates, domain boundaries, and element adjacency can be computed easily and very fast from the hierarchy, when needed. Data stored explicitly at tree elements can be reduced to pointers to the two possible children and information about local DOFs (for leaf elements). Furthermore, the hierarchical mesh structure directly leads to multilevel information which can be used by multilevel preconditioners and solvers.

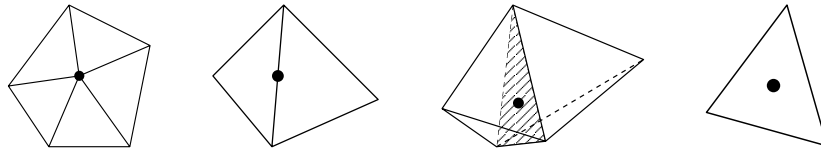
Access to mesh elements is available solely via routines which traverse the hierarchical trees; no direct access is possible. The traversal routines can give access to all tree elements, only to leaf elements, or to all elements which belong to a single hierarchy level (for a multilevel application, e.g.). In order to perform operations on visited elements, the traversal routines call a subroutine which is given to them as a parameter. Only such element information which is needed by the current operation is generated during the tree traversal.



## 4.2. 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  $S$  in local coordinates: Finite element functions on an element  $S$  are defined by a finite dimensional function space  $\bar{P}$  on a reference element  $\bar{S}$  and the (one to one) mapping  $\lambda^S : \bar{S} \rightarrow S$  from the reference element  $\bar{S}$  to the element  $S$ . In this situation the non vanishing basis functions on an arbitrary element are given by the set of basis functions of  $\bar{P}$  in local coordinates  $\lambda^S$ . Also, derivatives are given by the derivatives of basis functions on  $\bar{P}$  and derivatives of  $\lambda^S$ . The derivatives of  $\lambda^S$  solely depend on the geometry of  $S$ .

Each local basis function on  $S$  is uniquely connected to a global degree of freedom, which can be accessed from the element  $S$  via the DOF administration tool. ALBERT supports basis functions connected with DOFs, that are located at vertices of elements, at edges, at faces (in 3d), or in the interior of elements. DOFs at a vertex are shared by all elements which meet at this vertex, DOFs at an edge or face are shared by all elements which contain this edge or face, and DOFs inside an element are not shared with any other element. The support of the basis function connected with a DOF is the patch of all elements sharing this DOF, compare Figure 7.



**Figure 7.** Support of basis functions connected with a DOF at a vertex, edge, face (only in 3d), and the interior.

For a very general approach, we only need a vector of the basis functions (and its derivatives) on  $\bar{S}$  and a function for the communication with the DOF administration tool in order to access the degrees of freedom connected to local basis functions. By such information every finite element function (and its derivatives) is uniquely described on every element of the mesh.

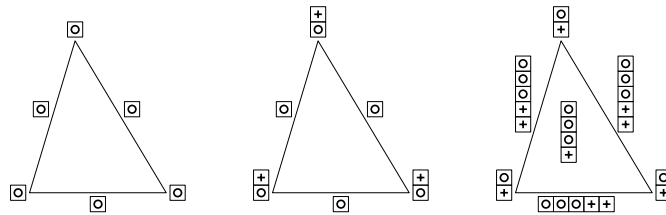
During mesh modifications, finite element functions must be transformed to the new finite element space. For example, a discrete solution on the old mesh yields a good initial guess for an iterative solver resulting in a smaller number of iterations for the solution of the discrete problem on the new mesh. Usually, these transformations can be realized by a sequence of local operations. Local interpolations and restrictions during refinement and coarsening of elements depend on the function space  $\bar{P}$  and the refinement of  $\bar{S}$  only. Thus, the subroutine for interpolation during an atomic mesh refinement is the efficient implementation of a representation of coarse grid functions by fine grid functions on  $\bar{S}$  and its refinement. A restriction during coarsening is implemented using similar information.

Lagrange finite element spaces up to order four are currently implemented in two and three dimensions. This includes the communication with the DOF administration as well as the interpolation and restriction routines.

### 4.3. Degrees of Freedom

Degrees of freedom (DOFs) connect finite element data with geometric information of a triangulation. For example, a continuous and piecewise linear finite element function can be described by the values of this function at all vertices of the triangulation. They build this function's degrees of freedom. A piecewise constant function is determined by its value in each element.

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. In Figure 8, three examples of DOF distributions for continuous finite elements in 2d are shown: piecewise quadratic finite elements  $\square$  (left), piecewise linear  $\square$  and piecewise quadratic  $\square$  finite elements (middle, Taylor-Hood element for Navier-Stokes: linear pressure and quadratic velocity), piecewise cubic  $\square$  and piecewise quartic  $\square$  finite elements (right, Taylor-Hood element for Navier-Stokes: quartic velocity and cubic pressure).



**Figure 8.** Examples of DOF distributions in 2d.

During adaptive refinement and coarsening of a triangulation, not only elements of the mesh are created and deleted, but also degrees of freedom together with them. The geometry is handled dynamically in a hierarchical binary tree structure, using pointers from parent elements to their children. For data corresponding to DOFs, which are usually involved with matrix-vector operations, simpler storage and access methods are more efficient. For that reason every DOF is realized just as an integer index, which can easily be used to access data from a vector or to build matrices that operate on vectors of DOF data. This results in a very efficient access during matrix/vector operations and in the possibility to use libraries for the solution of linear systems with a sparse system matrix ([18], e.g.).

Using this realization of DOFs two major problems arise:

- During refinement of the mesh, new DOFs are added, and additional indices are needed. The total range of used indices has to be enlarged. At the same time, all vectors and matrices that use these DOF indices have to be adjusted in size, too.

- During coarsening of the mesh, DOFs are deleted. In general, the deleted DOF is not the one which corresponds to the largest integer index. Holes with unused indices appear in the total range of used indices and we have to keep track of all used and unused indices.

These problems are solved by a general DOF administration tool. During refinement, it enlarges the ranges of indices, if no unused indices produced by a previous coarsening are available. During coarsening, a book-keeping about used and unused indices is done. In order to reestablish a contiguous range of used indices, a compression of DOFs can be performed; all DOFs are renumbered such that all unused indices are shifted to the end of the index range, thus removing holes of unused indices. Additionally, all vectors and matrices connected to these DOFs are adjusted correspondingly. After this process, vectors do not contain holes anymore and standard operations (BLAS, e.g.) can be applied.

In many cases, information stored in DOF vectors has to be adjusted to the new distribution of DOFs during mesh refinement and coarsening. Each DOF vector can provide pointers to subroutines that implements these operations on data (which usually strongly depend on the corresponding finite element basis). Providing such a pointer, a DOF vector will be transformed during mesh modifications.

All tasks of the DOF administration are performed automatically during refinement and coarsening for every kind and combination of finite elements defined on the mesh.

#### 4.4. Dimension Independent Program Development

Using the abstract definition of a finite element space with mesh, basis functions, and the DOF administration tool, a specific application can be implemented by providing problem dependent routines for assembling and solution of the discrete problems, as well as an error estimator/indicator. The adaptive method for finding a solution on a quasi-optimal mesh can then be performed by a black-box algorithm. The problem dependent routines are used for the calculation of discrete solutions on the current mesh and (local) error estimates. Here, the problem dependent routines heavily make use of library tools for assembling system matrices and right hand sides for an arbitrary finite element space, as well as tools for the solution of linear or nonlinear discrete problems. On the other hand, any specialized algorithm may be added if needed. The marking of mesh elements is based on general refinement and coarsening strategies relying on the local error indicators. During the following mesh modification step, DOF vectors (storing the coefficients of the discrete solution, e.g.) are automatically transformed to the new finite element spaces.

In the problem dependent routines only few parts of the finite element code depend on the dimension. Usually, all dimension dependent parts are hidden in the library. Hence, program development can be done in 2d, where execution is usually much faster and debugging is much easier (because of simple 2d visualization, e.g., which is much more involved in 3d). With no (or maybe few) additional changes,

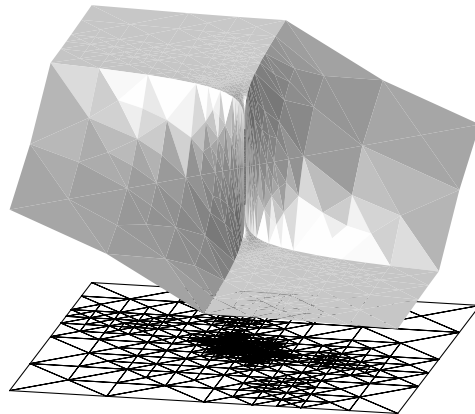
the program will then also work in 3d. This approach leads to a tremendous reduction of program development time for 3d problems.

## 5. APPLICATIONS

In this section we present three applications from research projects in order to demonstrate the flexibility of ALBERT. For further applications of ALBERT to CFD and dendritic growth we refer to [28].

### 5.1. Elliptic Problem with Discontinuous Coefficients

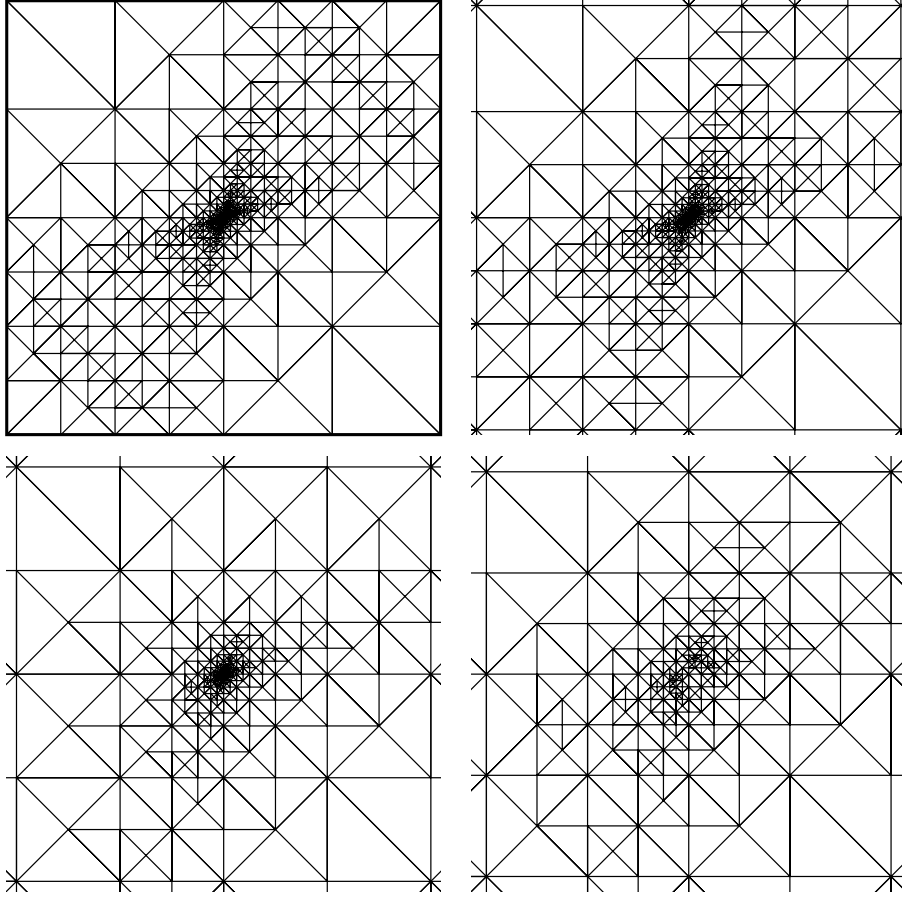
The first example is taken from [24]. The objective of this paper is the construction of a convergent adaptive algorithm for linear elliptic problems.



**Figure 9.** Graph of the discrete solution and underlying grid.

Ensuring a reduction rate of oscillation of the right hand side, together with an error reduction based on a posteriori error estimators, a simple and efficient adaptive FEM is constructed with a linear rate of convergence. Neither any preliminary mesh adaptation nor explicit knowledge of constants is needed for the adaptive procedure and any prescribed error tolerance is achieved in a finite number of steps. For testing the convergent algorithm in some worst case scenario, a solution to an elliptic equation with piecewise constant coefficients is constructed that is barely in  $H^1$  and behaves like  $r^{0.1}$  at the origin.

Due to the singularity, the grid is highly graded at the origin. It is worth to realize the strength of the singularity at hand in Figure 10. We see a mesh with  $< 2000$  nodes and three zooms at the origin, each obtained with a magnifying factor  $10^3$ , and yet exhibiting a rather strong grading. Without local mesh refinement it would not be possible to compute an approximation to the true solution with the same accuracy, even in 2d. The strong singularity is also reflected in Figure 9, which depicts the graph of the discrete solution over the underlying mesh.

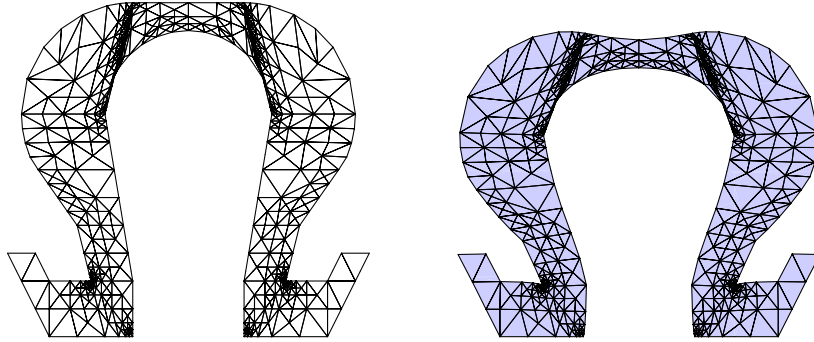


**Figure 10.** Final grid: full grid with  $< 2000$  nodes (upper left), zoom to  $(-10^{-3}, 10^{-3})^2$  (upper right), zoom to  $(-10^{-6}, 10^{-6})^2$  (lower left), and zoom to  $(-10^{-9}, 10^{-9})^2$  (lower right).

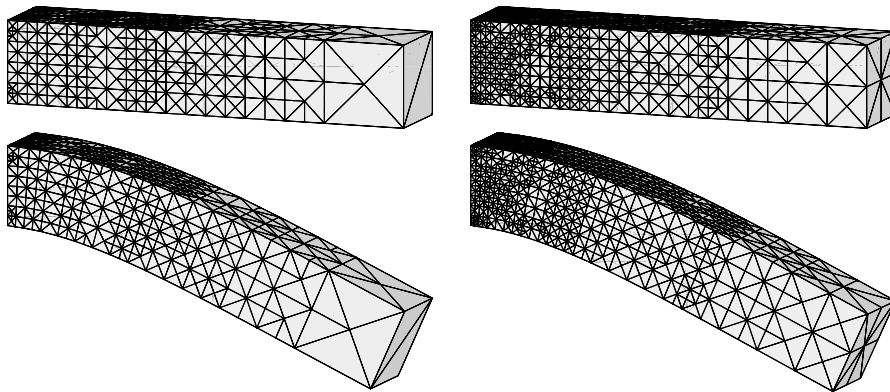
## 5.2. Linear Elasticity

We present here results from simulations of the linear elasticity problem in two and three dimensions. The unknown is the displacement vector, thus homogeneous Dirichlet boundary conditions imply no displacement from the reference location of a body (represented by the computational domain). Homogeneous Neumann boundary conditions imply that there are no normal stresses.

Figure 11 presents results for the model problem of a 2d body with ground contact, i.e. a homogeneous Dirichlet condition at the lower boundary (at  $y = 0$ ). At the other parts of the boundary, the homogeneous Neumann condition holds. Gravity pushes the body down, thus it becomes more flat.



**Figure 11.** An  $\Omega$ -shaped body under the influence of gravity in 2d. Adaptive grid and deformed  $\Omega$ .



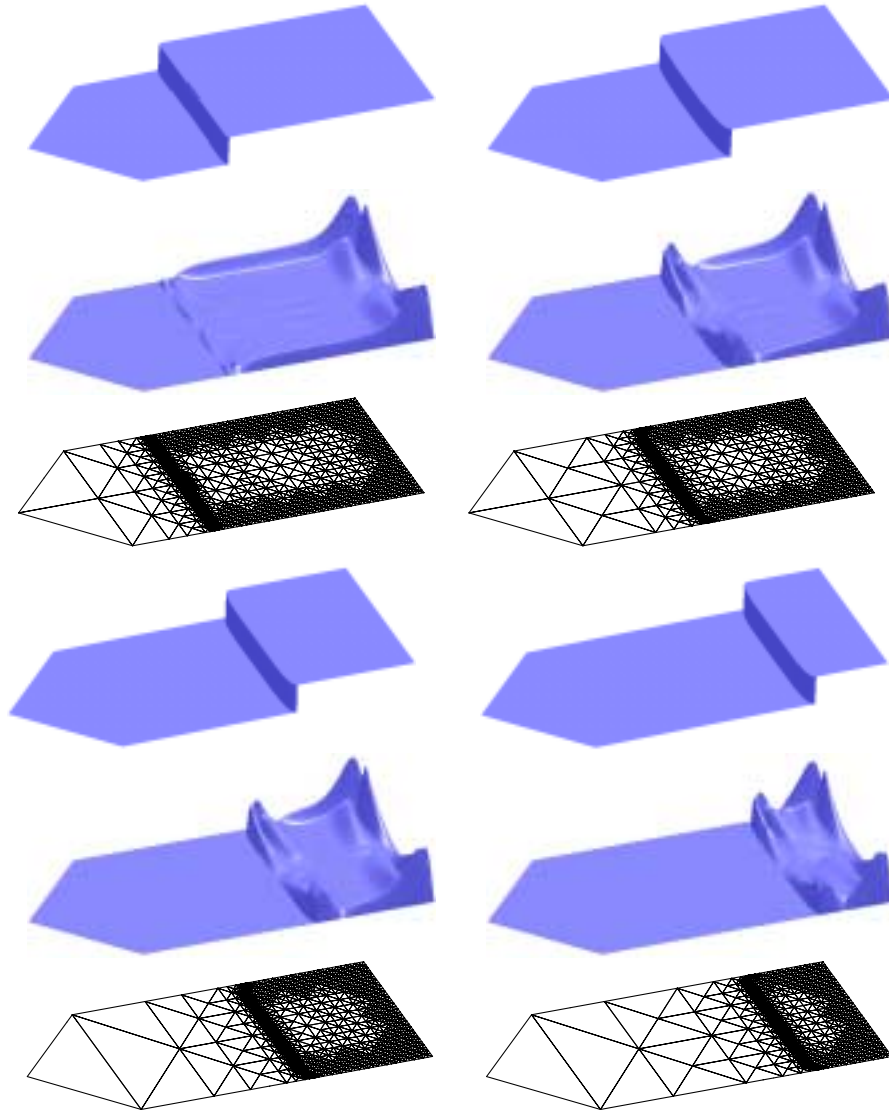
**Figure 12.** Deformation of a horizontal rod, clamped on left side, under the influence of gravity. Adaptive grids and deformed rod from two different adaptive refinements.

In Figure 12, we show adaptively refined meshes and computed deformations of a horizontal rod in three space dimensions, which is clamped on the left side (homogeneous Dirichlet boundary condition). No-stress conditions hold at all other boundaries. The force of gravity leads to a bending of the rod, with strongest stresses near the mounting.

Besides three additional entries in the symmetric system block matrix in 3d, the code for solving this problem is the same for 2d and 3d. Thus, program development was done in 2d. For the 3d code only these three additional entries, which have a similar structure as the entries for 2d, were added.

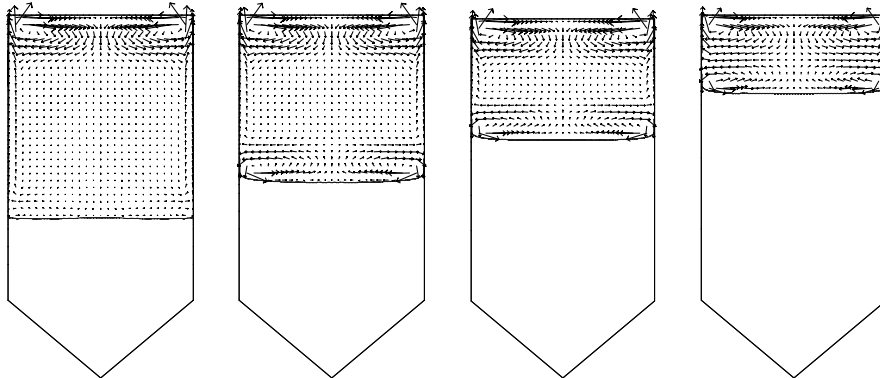
### 5.3. Phase Transition with Convection

The last application is a combination of phase transition and fluid flow [13, 14]. It models the industrial growth of a semiconductor crystal by the vertical Bridgman method with natural convection in the melt. The mathematical model couples the



**Figure 13.** Bridgman problem: Graphs of enthalpy and modulus of velocity over adaptive grids for four different times.

classical Stefan problem with the Navier-Stokes equations in the melt: 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 13 depicts the graphs of the enthalpy, modulus of the velocity over the adaptive grids and Figure 14 shows the isoline of the melting temperature and the velocity in the melt for different times of the



**Figure 14.** Bridgman problem: Isoline of the melting temperature and velocity in the melt for four different times.

simulation. The numerical approximation for the Navier-Stokes equations uses the  $P_3$ - $P_2$  Taylor-Hood element for the velocity and pressure. Piecewise linear finite elements are used for the discretization of the temperature and enthalpy in the Stefan problem.

Due to the discontinuity of the (true) enthalpy at the interface which is approximated by a continuous finite element function, it is clear that a high resolution of the mesh is needed near the interface. Since this interface moves in time not only mesh refinement but also mesh coarsening is needed in order to de-refine highly refined regions of the mesh when this high resolution is not needed anymore.

## 6. CONCLUSIONS

We described the main ideas which had an influence on the development and implementation of the finite element toolbox ALBERT. Due to the abstract definition of basis functions, use of quadrature formulas and the DOF administration tool only few parts of a finite element code depend on the dimension. Usually, these dimension-dependent parts are hidden in a library. This allows a dimension-independent programming of applications to the greatest possible extent.

ALBERT is freely distributed for research and education. Information about getting ALBERT can be found at

<http://www.mathematik.uni-freiburg.de/IAM/ALBERT>

and the detailed description of algorithms and data structures is given in [29].

## REFERENCES

1. Ainsworth M. and Oden J. T., *A unified approach to a posteriori error estimation using element residual methods*, Numer. Math. **65** (1993), 23–50.
2. Ainsworth M. and Senior B., *Aspects of an adaptive hp-finite element method: Adaptive strategy, conforming approximation and efficient solvers*, Comput. Methods Appl. Mech. Eng. **150** (1997), 65–87.



3. Ainsworth M., Senior B. and Andrews D., *Preconditioners for the adaptive hp version finite element method*, in The mathematics of finite elements and applications, Highlights 1996 (J. R. Whiteman, ed.), pp. 81–91, Chichester: Wiley, 1997; Proceedings of the 9th conference, MAFELAP 1996, Uxbridge, GB, June 25–28, 1996.
4. Babuška I. and Rheinboldt W., *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal. **15** (1978), 736–754.
5. Bank R. and Weisser A., *Some a posteriori error estimators for elliptical partial differential equations*, Math. Comput. **44** (1985), 283–301.
6. Bank R. E., *PLTMG: a software package for solving elliptic partial differential equations user's guide 8.0*, Software – Environments – Tools 5, Philadelphia, PA: SIAM. XII, 1998.
7. Bänsch E., *Local mesh refinement in 2 and 3 dimensions*, IMPACT Comput. Sci. Engrg. **3** (1991), 181–191.
8. Bänsch E. and Siebert K. G., *A posteriori error estimation for nonlinear problems by duality techniques*, Preprint 30, Universität Freiburg, 1995.
9. Bastian P., Birken K., Johannsen K., Lang S., Reichenberger V., Wieners C., Wittum G. and Wrobel C., *Parallel solution of partial differential equations with adaptive multigrid methods on unstructured grids*, in High performance computing in science and engineering '99 (E. Krause and et al., eds.), pp. 496–508, Berlin, Springer, 2000; Transactions of the High Performance Computing Center Stuttgart (HLRS), 2nd workshop, Stuttgart, Germany, October 4-6, 1999.
10. Beck R., Erdmann B. and Roitzsch R., *An object-oriented adaptive finite element code: Design issues and applications in hyperthermia treatment planning*, in Modern software tools for scientific computing (E. Arge and et al., eds.), pp. 105–124, Boston: Birkhaeuser, 1997; International workshop, Oslo, Norway, September 16–18, 1996.
11. Becker R. and Rannacher R., *A feed-back approach to error control in finite element methods: Basic analysis and examples*, East-West J. Numer. Math. **4** (1996), 237–264.
12. Bornemann F. A., Erdmann B. and Kornhuber R., *A posteriori error estimates for elliptic problems in two and three space dimensions*, SIAM J. Numer. Anal. **33** (1996), 1188–1204.
13. Boschert S., Schmidt A. and Siebert K. G., *Numerical simulation of crystal growth by the vertical Bridgman method*, in Modelling of Transport Phenomena in Crystal Growth (J. Szymd and K. Suzuki, eds.), Development in Heat Transfer Series, WIT Press, 2000.
14. Boschert S., Schmidt A., Siebert K. G., Bänsch E., Benz K., Dziuk G. and Kaiser T., *Simulation of industrial crystal growth by the vertical Bridgman method*, Report 00-11 ZeTeM Bremen and Preprint 14/2000 Freiburg, to appear.
15. Chen Z. and Nochetto R. H., *Residual type a posteriori error estimates for elliptic obstacle problems*, Numer. Math. **84** (2000), 527–548.
16. Ciarlet P. G., *The finite element method for elliptic problems*, North-Holland, 1987.
17. Demkowicz L., Oden J. T., Rachowicz W. and Hardy O., *Toward a universal h-p adaptive finite element strategy, Part 1 – Part 3*, Comp. Methods Appl. Mech. Engrg. **77** (1989), 79–212.
18. Dörfler W., *FORTTRAN-Bibliothek der Orthogonalen Fehler-Methoden*, Manual, Mathematische Fakultät Freiburg, 1995.
19. ———, *A convergent adaptive algorithm for Poisson's equation*, SIAM J. Numer. Anal. **33** (1996), 1106–1124.
20. Eriksson K. and Johnson C., *Adaptive finite element methods for parabolic problems I: A linear model problem*, SIAM J. Numer. Anal. **28** (1991), 43–77.
21. Fierro F. and Veiser A., *On the a posteriori error analysis for equations of prescribed mean curvature*, in preparation.
22. Kornhuber R., *Adaptive monotone multigrid methods for nonlinear variational problems*, Teubner, Stuttgart, 1997.
23. Kossaczky I., *A recursive approach to local mesh refinement in two and three dimensions*, J. Comput. Appl. Math. **55** (1994), 275–288.

24. Morin P., Nochetto R. H. and Siebert K. G., *Data oscillation and convergence of adaptive FEM*, SIAM J. Numer. Anal. **38** (2000), 466–488.
25. ———, *Local problems on stars: A posteriori error estimators, convergence, and performance*, Preprint 29/2000 Freiburg, 2000.
26. Nochetto R. H., Schmidt A. and Verdi C., *A posteriori error estimation and adaptivity for degenerate parabolic problems*, Math. Comp. **69** (2000), 1–24.
27. Nochetto R. H., Siebert K. G. and Veerer A., *Pointwise a posteriori error control for elliptic obstacle problems*, in preparation.
28. Schmidt A. and Siebert K. G., *Concepts of the finite element toolbox ALBERT*, Preprint 17/98, Freiburg, 1998; to appear in Notes on Numerical Fluid Mechanics.
29. ———, *ALBERT: An adaptive hierarchical finite element toolbox*, Preprint 06/2000, Freiburg, 2000; Documentation.
30. Veerer A., *Efficient and reliable a posteriori error estimators for elliptic obstacle problems*, Preprint 02/2000, Freiburg, 2000; to appear in SIAM J. Numer. Anal.
31. Verfürth R., *A posteriori error estimation and adaptive mesh-refinement techniques*, J. Comp. Appl. Math. **50** (1994), 67–83.
32. ———, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Wiley-Teubner, 1996.

A. Schmidt, Zentrum für Technomathematik, Fachbereich 3 Mathematik und Informatik, Universität Bremen, Postfach 33 04 40, D-28334 Bremen, Germany,  
*e-mail*: Schmidt@math.uni-bremen.de

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