INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **28**: 679–702 (1998)

A UNIFIED MESH REFINEMENT METHOD WITH APPLICATIONS TO POROUS MEDIA FLOW

ERIK J. HOLM[a,1] AND HANS PETTER LANGTANGEN[b,*]

[a] *Institute for Energy Technology (IFE), 2007 Kjeller, Norway*
[b] *Department of Mathematics, University of Oslo, PO Box 1053 Blindern, N-0316 Oslo, Norway*

SUMMARY

A unified algorithm is presented for the refinement of finite element meshes consisting of tensor product Lagrange elements in any number of space dimensions. The method leads to repeatedly refined n-irregular grids with associated constraint equations. Through an object-oriented implementation existing solvers can be extended to handle mesh refinements without modifying the implementation of the finite element equations. Various versions of the refinement procedure are investigated in a porous media flow problem involving singularities around wells. A domain decomposition-type finite element method is also proposed based on the refinement technique. This method is applied to flow in heterogeneous porous media. © 1998 John Wiley & Sons, Ltd.

KEY WORDS: finite elements; h-refinement; porous media flow; object-oriented programming

1. INTRODUCTION

Adaptive grids constitute an important tool for increasing the computational efficiency of finite element solvers. There are three basic approaches to mesh refinement: decreasing the element size (h-refinement), increasing the polynomial order of the finite element basis functions (p-refinement), and moving the nodes (r-refinement). Combinations of these methods are also possible. In the present paper the focus is on general 1D, 2D and 3D mesh refinement techniques that can be implemented in existing finite element codes without, at least in principle, any modifications of the standard assembly process and the computation of elemental matrices and vectors. This constraint is very difficult to fulfil when applying p-refinement [1]. The flexibility of r-refinement is somewhat limited. Therefore, the focus is on h-refinement. An excellent overview of h-refinement procedures, including mesh generation, error estimation and practical experience, is provided by Strouboulis and Haque [2].

There are two types of h-refinement, mesh regeneration or element subdivision. In the latter approach an element-wise refinement indicator is used to determine whether an element needs to be refined (or coarsened). Mesh regeneration means that the whole finite element grid is recomputed and element sizes are distributed according to a density function [1,3]. The method proposed in this paper is founded on element subdivision, but it also shows some similarities to mesh regeneration.

* Correspondence to: Department of Mathematics, University of Oslo, PO Box 1053 Blindern, N-0316 Oslo, Norway. E-mail: hpl@math.uio.no
[1] E-mail: erikh@ife.no

CCC 0271–2091/98/140679–24$17.50

Received April 1997
Revised August 1997

When an element is to be divided into new elements, several approaches are possible. If the original mesh consists of quadrilateral or hexahedral elements, subdivision into new elements of the same type leads to irregular nodes (also called slave nodes, constrained nodes or hanging nodes) on a side if the neighbor element along that side is not also refined. Inserting triangles or tetrahedra into the neighboring element can transform the irregular node to a regular one, otherwise an additional constraint equation must be applied for the irregular node. Finite element grids composed solely of triangles or tetrahedra can be refined in a way that preserves the element type and avoids the introduction of irregular nodes [4,5]. The latter type of algorithms might be rather complicated in 3D. Simple, compact algorithms can be developed by working with quadrilateral and brick elements alone, and allowing irregular nodes. Moreover, these algorithms give the user better control and more flexibility. This strategy is therefore adopted in the present work.

Adequate element-wise refinement indicators can, in the simplest case, be based purely on geometric considerations. For example, in many problems certain properties of the solution, such as singularities, abrupt changes in coefficients, boundary layers or sharp fronts, are known beforehand or are found by a simple inspection of the computed solution. Critical parts of the domain can then easily be established. These type of indicators are referred to as geometric refinement indicators. They have been successfully used in numerous flow problems. A more general and mathematically appealing approach is to relate the refinement indicator to some kind of *a posteriori* error estimation, based on the computed solution with the current mesh. Such error estimation makes it possible to design optimal grids with uniform error distribution. Furthermore, it gives the opportunity to *control* the error.

The main goal of this study is to develop a unified algorithm that works in 1D, 2D and 3D (and even higher space dimensions) for structured as well as unstructured grids. Such unified algorithms tend to be much more compact and simple than more geometrically intuitive approaches. Existing 3D algorithms usually involve lots of special treatments of various geometric refinement situations. This complicates the verification of the associated software. It is thought that software reliability is significantly increased by adopting simple, unified algorithms that treat the number of space dimensions merely as a parameter. This has proven to be possible if the elements are of tensor product type. The present refinement approach has in fact been successfully integrated in finite element codes that parameterize the number of space dimensions, with a resulting dramatic decrease of the development time for 3D applications. Trangenstein [6] describes a unified adaptive mesh refinement algorithm for any number of space dimensions, but the algorithm and the associated data structures were tailored to explicit time stepping, finite difference/volume methods and patch refinement (see below). Therefore, both the refinement strategy and the constraint equations are not as general and flexible as aimed at in the present paper.

Numerous contributions in the literature describe refinement algorithms for quadrilateral and brick elements. Demkowicz *et al.* [7] gave a thorough description of a 2D algorithm for h–p-refinement and corresponding data structures. The irregular node constraints were handled by modifying the finite element basis functions, and hence also the computation of the elemental matrices and vectors. This is a common strategy, but in the present study a h-refinement algorithm is sought, in which the element-wise computations can be kept in traditional form such that adaptive grids can be easily incorporated into existing solvers without modifying well-tested software.

Most of the literature contributions deal only with 2D algorithms. Several important three-dimensional issues have been treated by Chellamuthu and Ida [8]. Their paper is referred to for further discussion of previous work. Morton *et al.* [9] presented a 3D algorithm in which

the constraints associated with irregular nodes are incorporated by modifying the finite element basis functions, i.e. they introduce a family of new transition elements. According to Morton *et al*., this approach is easier to integrate into an existing finite element code compared with methods based on separate constraint equations. However, the disadvantage with special transition elements [9,10] is that these new elements must be incorporated into the finite element library of a code, and the grid must then be composed of different element types. The present approach allows the original, standard element type and a standard, often highly optimized, finite element solver to be used.

Refinement algorithms that involve irregular nodes frequently employ the so-called one-level rule [8]. This results in only one constrained node on the boundary between elements with different refinement levels. Grids which are refined according to the one-level rule are usually referred to as one-irregular meshes. The advantage of one-irregular meshes is that the element size is enforced to vary more smoothly. However, the one-level rule allows only one subdivision at a time. For increased efficiency and flexibility, the present algorithm allows n-irregular grids. Nested grids can be easily obtained by repeated application of the n-irregular subdivision procedure. The construction of an n-irregular mesh allows each coarse grid element to be refined independently. In finite difference/volume methods it has been popular to apply n-irregular grids, usually referred to as patch refinements [6,11,12], especially for hyperbolic equations solved by explicit time stepping. Also in finite element methods for 2D single-phase porous media flow n-irregular meshes have proven to be successful [13,14]. In the present paper some advantages with n-irregular meshes are demonstrated for $n > 1$. The major part of the literature on one-irregular finite element grids is concerned with bilinear and trilinear elements, whereas the present algorithm is capable of handling tensor product Lagrange elements of any polynomial degree. Nine-node 2D Lagrange elements are treated in References [2,15], while the algorithm by Demkowicz *et al.* [7] is concerne d with midside nodes in the h–p-refinement.

A general approach for marking boundary conditions in a finite element grid is presented, and a special strategy is developed for new elements and nodes in the refined mesh to inherit the proper boundary condition information from the previous refinement level. This is an important topic that, from a general point of view, is not well-addressed in the literature. Essential boundary conditions are enforced exactly here, in contrast to the simpler, but frequently used, penalty methods [7]. Moreover, the construction of the constraint equations for repeatedly constructed n-irregular meshes is described. Contrary to Chellamuthu and Ida [8], where different algorithms must be used for constrained nodes on edges and faces of a 3D element, a unified constraint detection and construction algorithm that works in 2D, 3D and higher dimensions is presented. The importance of flux continuity is also described, in addition to continuity of the primary unknown, and the method in which this can be achieved by a symmetrization procedure of the standard constraint equations.

Object-oriented implementation techniques seem to be very advantageous in the refinement process [6,16]. This is also the case in the present study. Liu *et al*. [17] discussed object-oriented implementation techniques for one-irregular 2D refinement of finite element grids. In the present paper, another important feature of object-orientation is outlined: by treating the whole grid as an object, and applying object-oriented design of the whole finite element simulator, the proposed algorithm allows existing simulators to apply the new adaptive mesh refinement tools with almost no modification of the original code. A brief account of the ideas of such an object-oriented design is given.

Finally, a novel application is described for the n-irregular mesh refinement technique for a certain domain decomposition-type method, where each subproblem has a grid that actually

covers the whole domain. The approach is an improvement of the global–local analysis method [18,19]. The numerical examples concern single-phase flow in heterogeneous porous media.

The paper is organized as follows. In Section 2 the refinement method and its implementation are described. Section 3 presents a test problem from porous media flow and the refinement indicators to be used. Section 4 deals with numerical experiments. The use of the refinement method in a certain domain decomposition-type method is presented and tested in Section 5, and concluding remarks are given in Section 6.

2. THE GRID REFINEMENT ALGORITHM

The grid refinement algorithm is restricted to finite elements of (deformed) hypercube shape, but it can handle higher-order basis functions and any number of space dimensions. The algorithm works in an element-by-element fashion, where the subdivision of one element is independent of the others. Only some information about whether neighbouring elements are refined or not is required when forming the constraint equations. The refinements can be repeated to produce nested grids. The details of the algorithm at a given refinement level are described below.

2.1. Refinement of a single element

Consider a general d-dimensional finite element mesh consisting of hypercube elements. These elements are later referred to as the parent elements. Each parent element can be mapped onto a reference element $[-1, 1]^d$ in a co-ordinate system where the co-ordinates are denoted by ξ. Let the corresponding physical co-ordinates be x, with the relation $x = M(\xi)$. Normally, the mapping M is expressed in terms of the basis functions $N_i(\xi_1, \ldots, \xi_d)$ on the reference element. Assume that the elements with element numbers in the integer set $R = \{e_1, \ldots, e_E\}$ have been chosen for refinement. For each $e_i \in R$, the element is refined by partitioning the reference element $[-1, 1]^d$ into a regular, uniform grid with grid points

$$\xi_{i_1,\ldots,i_d}, \qquad i_j = 1, \ldots, m_j + 1, \qquad j = 1, \ldots, d.$$

For n-irregular grids, $m_j = n + 1$. This grid is referred to as the prototype grid of parent element e_i. Anisotropic refinement is easily accomplished, but omitted here to simplify the notation. New elements with nodes $M(\xi_{i_1}, \ldots, _{i_d})$ are easily constructed from the prototype grid. These new nodes are called child nodes. For example, if the original elements are of multilinear type, $\Pi_{j=1}^{d} m_j$ new multilinear child elements are defined. In the case of multiquadratic elements, $2^{-d}\Pi_{j=1}^{d} m_j$ new child elements can be created.

The child nodes and the nodes in the parent mesh make up the new refined mesh. Assuming that the m_j parameters, $j = 1, \ldots, d$, are the same for all of the refined elements, all the child nodes of a parent element become regular nodes in the refined mesh if a proper subset of the neighboring parent elements are also refined. However, if a neighboring element is not refined, the child nodes at the boundary between the refined and non-refined parent elements become irregular nodes. Figure 1 depicts a situation with both regular and irregular child nodes at the boundary of a refined parent element. Note that all interior child nodes of a refined parent element become regular nodes in the new refined mesh.

The value of a finite element field at an irregular node is constrained by the value at the neighboring regular nodes. Let the parent element have basis functions $N_i(\xi)$ and nodal values

α_i, $i \in J$, where J is an appropriate set of indices. The constraint equation to be imposed at an irregular node ξ_k is

$$\sum_{i \in J} N_i(\xi_k)\alpha_i = \alpha_k, \tag{1}$$

where α_k ($k \notin J$) is the unknown value of the finite element field at the irregular node. During the generation of the child nodes, all child nodes on the boundary of the parent element are marked as candidates for constraints. A postprocessing procedure determines which of these candidates are irregular nodes in the final grid.

In practice, there are two main difficulties in the present grid refinement algorithm; determination of the constraint equations and assignment of boundary conditions to new nodes on the boundary. These topics are treated in the next two subsections.

2.2. Determining the constraint equations

Child nodes marked for constraints during the construction of the refined grid may become regular nodes if a proper set of neighboring elements are also refined. To determine the real irregular nodes, constraining elements are sought, i.e. elements that contain a node which is not coinciding with the standard nodes in the element. In this general approach, the possibilities of repeated refinements and of more than one irregular node at the element boundary must be considered. A constraining element is sought through all possible element generations by comparing the candidates parents, grandparents, etc. with the nodes of adjacent elements. If a constraining element is found, the node is activated as an irregular node in the grid.

To illustrate the algorithmic details, a one-level refinement of a uniform grid is studied (see Figure 2). Here, element 2 has a non-standard node A. Element 2 is then referred to as a constraining element for the irregular node A. In the construction algorithm, node A is marked for further investigation because it was generated on the boundary of its parent element. The constraint equation for node A is instantly constructed using the finite element interpolation functions in element 2 according to Equation (1). Nodes B and C are then referred to as the parent nodes of A (the other two nodes of element 2 do not contribute to the constraint

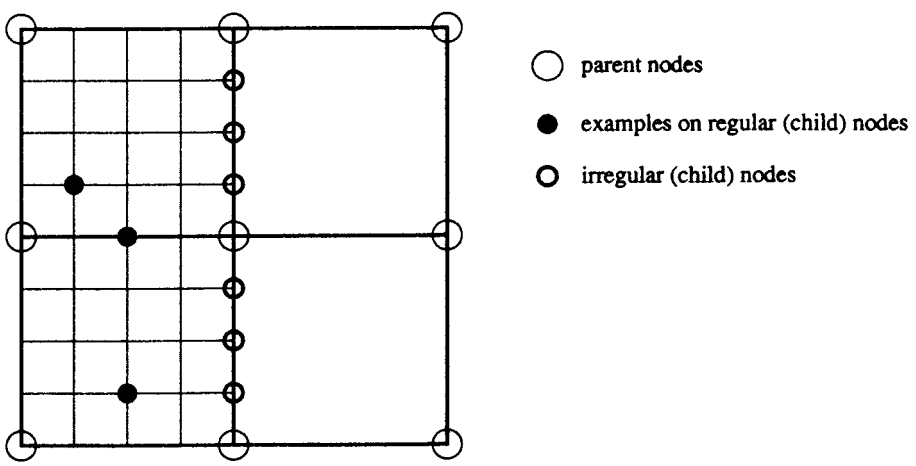

Figure 1. Sketch of mesh refinement.

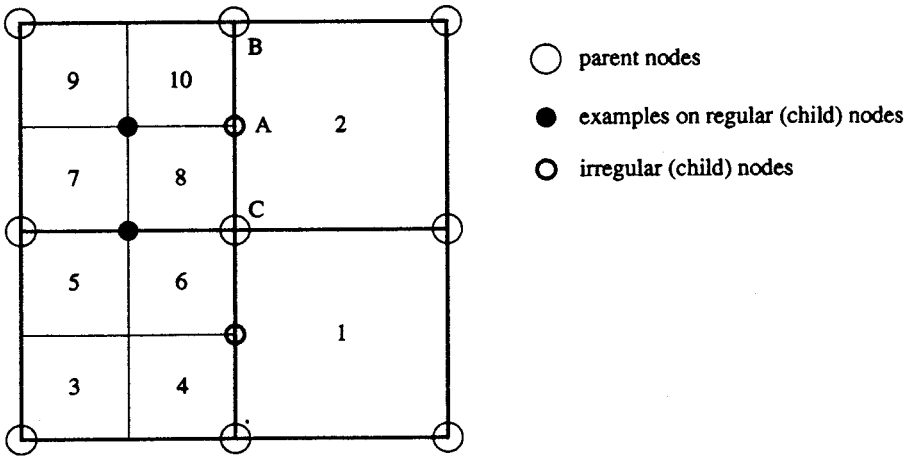

Figure 2. Sketch of a one-level mesh refinement, with element numbers indicated.

equation because the interpolation functions associated with these nodes vanish at node A). In this one-level refinement, it is easy to realize that a constraining element for node A must include node B and C among its standard nodes. A check-list of nodes consisting of the parent nodes of A is established. Searching through the adjacent elements of the irregular node candidate A, it is clear that element 2 contains both nodes in the check list, but not A itself. Therefore, node A fulfils the requirements of being a true irregular node, and element 2 is the constraining element for node A.

The search for a constraining element for a possible irregular node is somewhat more complicated in a grid with several levels of refinement. Constraining elements can now originate from any of the levels prior to the current one. First, the previous level is examined as before, using a check-list containing parent nodes. In Figure 3 the constraint candidate node D is investigated by searching for its parent nodes A and B among elements in the neighborhood (elements 2, 8, 11 and 13). A constraining element is not found, i.e. an element

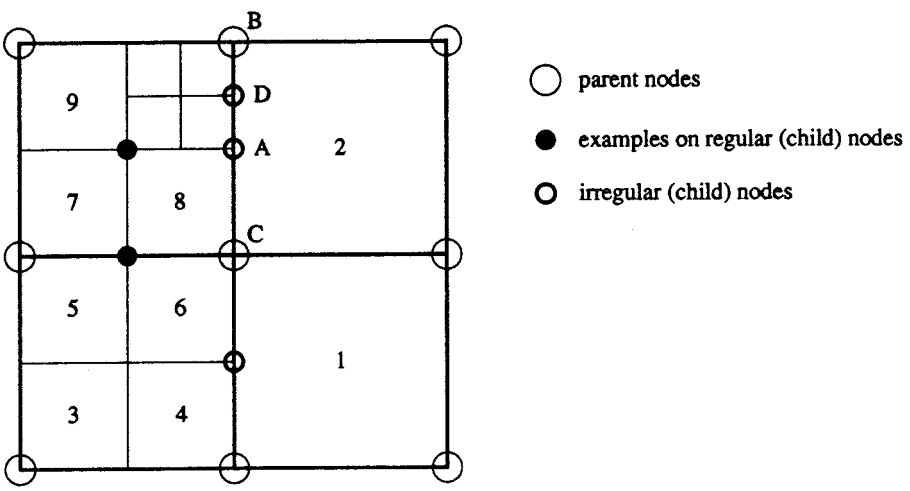

Figure 3. Sketch of a two-level mesh refinement.

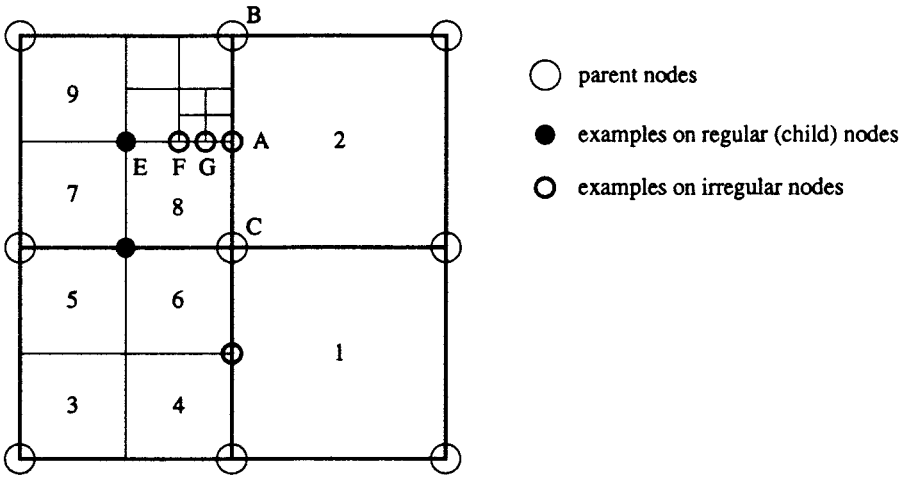

Figure 4. Sketch of a three-level mesh refinement.

containing A and B, but the parent A itself is a constraint candidate with corresponding grandparents C and B. A constraining element from the grandparent generation will clearly include the nodes B and C. Generally, a new check-list is established for the next level of refinement, where the youngest constrained nodes are substituted in the check-list by their parents, and finally, the duplicated members are removed. The algorithm is implemented as a recursively called function. The recursion is terminated when either a constraining element is found, or when all nodes in the check-list are established as truly regular. In the present example, the second check-list level reveals a constraining element, namely element 2.

To see the necessity of unwrapping only the newest generation at each step, consider the situation in Figure 4 and the constraint candidate node G. A constraining element containing the parents F and A is not found. Substituting the constrained parent F with its parents E and A in the check-list, the constraining nodes E and A are detected in element 8. Node G is then truly irregular. Note that if the older constrained parent A was simultaneously substituted by its parents B and C, the search would have been for an element with all of the nodes B, C and E, and would have failed.

This algorithm has proven to be robust for any number of space dimensions, any order of the interpolation functions, any number of repeated refinements, and any value of n. It is important to note that after constraining elements for a node, the search can be restricted to elements evolving from neighbors of the nodes of the parent element in the previous grid. The amount of work involved in the search algorithm is very small compared with operations involving the total number of nodes (such as the solution process). This also makes it easier to design parallel versions of the refinement method. The present algorithm for construction of constraint equations is far more general and free of special considerations than grid generation schemes, in which new neighbor elements are introduced to transform irregular nodes to regular nodes. This advantage is especially important in 3D and for higher-order elements.

2.3. Incorporating the constraint equations

The traditional finite element equations, computed in the standard element-by-element fashion without incorporating any constraints due to irregular nodes, can be written as

$$\sum_{j=1}^{n} A_{ij}\alpha_j = b_i, \quad i = 1, \ldots, n, \tag{2}$$

where n is the number of regular and irregular nodes. It is assumed that a single, scalar differential equation is solved. Vector equations are easily handled by applying the method to each component equation. The constraint equations are all of the form of Equation (1). This can be rewritten as

$$\sum_{j=1}^{n} C_{kj}\alpha_j = \alpha_k, \tag{3}$$

if node k is constrained. The constants C_{kj} are zero for the degrees of freedom that do not enter the constraints.

The solution of Equation (2) is subject to the linear constraints of Equation (3) and is accomplished by inserting Equation (3) into (2)

$$\sum_{\substack{j=1 \\ j\neq k}}^{n} (A_{ij} + C_{kj}A_{ik})\alpha_j = b_i, \quad i = 1, \ldots, n, \quad i \neq k. \tag{4}$$

This modified equation system incorporates the single constraint equation (3), but suffers from two problems. First, the modification destroys any initial symmetry in the coefficient matrix. Second, for several differential operators, the solution must be continuous at the nodes and the associated flux must be continuous in a weak sense over the element boundaries, but this is not generally fulfilled in the system (4). Both problems can solved by the following modification of the original system

$$\sum_{\substack{j=1 \\ j\neq k}}^{n} [A_{ij} + C_{kj}A_{ik} + C_{ki}(A_{kj} + C_{kj}A_{kk})]\alpha_j = b_i + C_{ki}b_k, \quad i = 1, \ldots, n. \tag{5}$$

By inspection, the new coefficient matrix is symmetric if $A_{ji} = A_{ij}$. Appendix A shows that Equation (5) preserves weak continuity of the flux. Symmetric introduction of constraints is also ensured by the methods in References [2,7], although limited to one-irregular grids. Incorporation of a set of constraints is trivially accomplished by applying Equation (5) for each constraint equation.

The modification (5) is performed globally after the elemental contributions are assembled in the standard manner. This means that the computation of the elemental matrices and vectors is unaffected by the grid refinement procedure.

2.4. Boundary conditions at child nodes

It is assumed that boundary nodes in the parent mesh are marked with one or several boundary indicators. Each indicator can be on or off at the nodes. A boundary indicator can be used to e.g. select the nodes that are subject to a particular essential boundary condition in the grid. The boundary indicator concept is general and can also be used to mark internal boundaries or regions. Child nodes must, of course, inherit the proper boundary indicators. This is a non-trivial problem, the solution of which is presented below.

A child node that coincides with a parent node can simply inherit the boundary indicators of the parent node. A child node on an edge or side of an element inherits a boundary indicator if all parent nodes on the edge or side are subject to this indicator. To avoid complicated testing procedures and handling of many special cases, especially in 3D grids a compact and general algorithm was developed for inheriting boundary indicators. This algorithm applies to tensor product elements of Lagrange type in any number of space dimensions.

Child nodes that are generated in a parent element, where at least one boundary indicator is present in at least one parent node, may be subject to boundary indicators and must be investigated further. For each such child node, with co-ordinate y, we compute the quantity

$$U(y, k) = \sum_{\ell} W_k(\ell) N_\ell(y), \quad k = 1, \ldots, n_b. \tag{6}$$

Here, k is a boundary indicator number, n_b is the total number of possible boundary indicators, N_ℓ denotes the basis functions in the parent element, and $W_k(\ell)$ is a weight function defined by $W_k(\ell) = k$ if parent node ℓ is subject to boundary indicator k, otherwise $W_k(\ell) = 0$. Let V_1 denote the parent element domain, let V_2 be a side, and let V_3 be an edge. We write $y \in V_j$, $j = 1$, 2, 3, when y is a point in V_j. Similarly $\ell \in V_j$ means that parent node ℓ lies in V_j. The following result is given for tensor product elements of Lagrange type

$$\sum_{\ell \in V_j} N_\ell(y) = 1, \quad y \in V_j, \quad j = 1, 2, 3. \tag{7}$$

Generalization of this result to any number of space dimensions is straightforward. With the result of Equation (7) and the expression (6) it can be determined whether a child node with co-ordinates y is subject to boundary indicator k by testing if $U(y, k) = k$. Inheritance of boundary indicators is hence reduced to evaluating a simple formula.

2.5. Object-oriented implementation

The dynamic data management and the often rather complicated algorithms that are needed in adaptive grid refinement are most conveniently implemented in a programming language that supports abstract data types and object-oriented programming (OOP). Most of the literature on data structures and implementation of grid refinement are, nevertheless, described in terms of primitive FORTRAN arrays, although the use of abstract data types and OOP, especially in C++, is emerging [6,16,17]. A thorough treatment of OOP, C++, and adaptive finite element meshes is presented in Reference [17] (although only 2D one-irregular refinements are considered). The programming efforts are greatly reduced and simplified by employing the modern software development techniques, but computational efficiency must not be abandoned in favor of more advanced and appealing data types that might yield very elegant code. The general rule of thumb seems to be to use OOP for high-level data and algorithm management, and use 'do-loops' and primitive array structures in CPU-intensive low-level code. The current refinement algorithm has been implemented in C++, where a compromise between primitive array structures and more high-level data types was sought. The details will not be discussed further in this paper. Interested readers might consult Trangenstein [6], Lemke *et al.* [16], Liu *et al.* [17], and Ewing [20], and the references therein for implementation details of other, but related, refinement strategies. Instead, the use of object-oriented programming to integrate the whole grid refinement process in an existing finite element code is studied here. Using OOP at such a high abstraction level in the program usually allows most of the convenient OOP constructs to be applied without sacrificing computational efficiency [21,22].

The grid refinement procedure has been implemented in the software system Diffpack [23]. Diffpack provides the basic building blocks for finite element codes in terms of a library of C++ classes. These classes reflect mathematical quantities and numerical methods at various abstraction levels, ranging from simple array structures to linear system solvers and finite element fields. Object-oriented design has been a vital mean for structuring the library and simplifying the application code [24], yet with a high efficiency [21]. For example, to solve a Poisson equation or a standard elasticity problem in Diffpack, the programmer only needs to provide the

integrands in the weak residual form, essential boundary conditions, some problem specific I/O, and a high-level program flow routine.

Finite element grids are represented by an object of class GridFE in Diffpack. This class contains standard data structures for nodal co-ordinates, element connectivity and boundary information. However, the users of a class never operate directly on the data structures, but through member functions of the class. The member functions define a mathematical interface to grids, i.e. they define common grid operations, such as looking up the co-ordinates of a nodal point and looking up the global degree of freedom number corresponding to the local node number in a particular element. If the internal data structures are changed, e.g. by replacing arrays by lists or trees, the member function interface remains the same, and the code in other parts of the program that use class GridFE is not affected. This encapsulation technique is particularly advantageous when incorporating an adaptive mesh class in an existing library.

The name Grid2FE is used here to denote the new adaptive mesh class. Naturally, Grid2FE has an internal data management that is much more complicated than GridFE. Nevertheless, the mathematical interface (i.e. the member functions) is almost the same for a standard grid as for an adaptive grid. Since all the data structures are hidden inside the class, rather than appearing as individual arrays in long argument lists as in FORTRAN subroutine libraries, both standard grids and adaptive grids are transferred to other routines of the same argument type. If Grid2FE is a so-called subclass [25] of GridFE, OOP techniques allow the application code to be written solely using the name GridFE, except when declaring and allocating the grid object. The compiler will generate information such that at run-time the program knows whether the apparent GridFE object is actually a Grid2FE object or a standard GridFE object. In C++ this is technically achieved through the use of inheritance and virtual functions.

The finite element software that operates on GridFE objects needs standard GridFE functionality as well as information about the constraints. It is then natural to define a virtual function that extracts the constraint information. This function is of course empty in class GridFE, whereas class Grid2FE provides a version that extracts suitable data structures. Since the Diffpack finite element classes operate only on GridFE objects (or rather GridFE pointers or references), the existing library will also work without modifications when the more complicated Grid2FE objects for adaptive refinements are fed into the library computations. Similarly, application codes which use the library to solve initial-boundary value problems are also principally unaffected of whether the grid object is of standard or adaptive type. Of course, the application code will usually need some additional information on refinement strategies, etc. Again, this information is provided in terms of a class object, resulting in an order of ten new code lines in a simulator to incorporate the adaptive grid.

It is emphasized that many of the algorithmic decisions in the present work, e.g. the use of separate constraint equations rather than transition elements [9,10], are influenced by the object-oriented design and the corresponding possibility to incorporate the new Grid2FE in existing simulators without modification of already debugged implementation of the weak forms of the partial differential equations.

3. MODEL PROBLEM AND REFINEMENT INDICATORS

3.1. Boundary-value problem

Theory and numerical examples in this paper mostly refer to a model problem describing single-phase fluid flow in heterogeneous porous media. Let p be the fluid pressure, ϱg the specific fluid weight, ∇D the unit vector in the vertical direction, μ the dynamic viscosity, x a spatial point, x_r the location of a well, $r = 1, \ldots, n_w$, Q_r the corresponding well strength of a Dirac delta

function well model ($Q_r\delta(x-x_r)$), and finally, let K be the permeability. For constant ϱg, the fluid potential $\psi = p - \varrho g D$ is introduced as the primary unknown. Then, ψ is governed by

$$\nabla\cdot\left[\frac{K}{\mu}\nabla\psi\right] = \sum_{r=1}^{n_w} Q_r\delta(x-x_r) \quad \text{in } \Omega, \tag{8}$$

with the associated boundary condition

$$\nabla\psi\cdot n = 0 \tag{9}$$

on $\partial\Omega$, where n is the outward unit normal of $\partial\Omega$. Global mass conservation and Equation (9) imply the constraint $\Sigma_r Q_r = 0$ on the well strengths.

The bilinear forms $a(u, v)$, $a_j(u, v)$: $V\times V\to\mathbb{R}$ are defined by

$$a(u, v) = \sum_{j=1}^{n_e} a_j(u, v), \qquad a_j(u, v) = \int_{\Omega_j} \nabla u\cdot K\cdot\nabla v \, d\Omega, \quad u, v\in V,$$

where V is the Sobolov space $H^1(\Omega)$, Ω_j denotes element j, and n_e is the total number of elements. Also, define the linear functional

$$L(v) = \sum_{i=1}^{n_w} Q_r v(x-x_r), \quad v\in V.$$

Introducing a finite dimensional subspace $V_h\subset V$, and letting $\hat{\psi} = \Sigma_{i=1}^{n} N_i\hat{\psi}_i\in V_h$ be the finite element approximation to ψ, gives the standard discrete weak form for this problem: Find $\hat{\psi}\in V_h$ such that

$$\sum_{j=1}^{n} a(N_i, N_j)\hat{\psi}_j = L(N_i), \quad i = 1, \ldots, n.$$

3.2. Geometric refinement indicators

Simple refinement indicators based on geometric properties of the solution or the coefficients in the PDE can be used as an alternative to, or in combination with, *a posteriori* error estimators. *A posteriori* estimators must often be specially designed for the current problem, and may not always be robust enough to justify the resulting overhead [14]. Geometric error indicators are natural in many problems. For example, in porous media flow, it is known that the solution has singularities at the wells and at corners of the domain (with interior angle larger than π). Moreover, a h-refinement is desired in the vicinity of large jumps in the permeability. Saturation equations in immiscible flow or concentration equations in miscible flow develop sharp fronts that are candidates for local mesh refinements. A trivial geometric indicator can then be based on the gradient of the solution. The advantage of geometric indicators is that they are very cheap to compute, easy to use, and give the user a good control over where the refinements are performed. True error estimators often tend to predict refinement in areas that later must be coarsened, but geometric indicators usually avoid the need for coarsening. In time-dependent problems it might be important to avoid coarsening to increase the computational efficiency [20]. The disadvantage with geometric refinement indicators is that the user has no direct control of the error in the computation.

3.3. Error estimation

Most of the state-of-the-art error estimators for problems involving the Laplace operator and smooth solutions have been investigated and compared by Babuska *et al.* [26]. They conclude that a discrete version of the improved Zienkiewicz–Zhu estimators [27,28] is the best estimator. The estimator used in the present study is a variant of the Zienkiewicz–Zhu (ZZ) estimator [1], and it should be noted that the solution exhibits a singularity and, in fact, infinite values in the wells.

Define the error $e = \psi - \hat{\psi}$. The gradient of ψ is often of particular physical interest, such as in the present model problem, where $(K/\mu)\nabla\psi$ is the fluid velocity. It is therefore appropriate to study e in the energy norm

$$\|e\|_{E(\Omega)} \equiv \sqrt{a(e,e)}, \qquad \|e\|_{E(\Omega_j)} \equiv \sqrt{a_j(e,e)}. \tag{10}$$

An error indicator is introduced for element j,

$$I_j = \frac{\sqrt{n_e}}{\epsilon} \frac{\|e\|_{E(\Omega_j)}}{\|e\|_{E(\Omega)}},$$

that measures the ratio of the error in the current element and a target mean square element error, including an adjustable factor ϵ. If $I_j > 1$, the error in element j is larger than the target error and the element should be refined. The following error measure can also be defined [1]: $\alpha_j = \|e\|_{E(\Omega_j)} / \sqrt{\int_{\Omega_j} \mathrm{d}\Omega}$.

The critical step now is to find an *a posteriori* estimate of $\nabla\psi$. The simple and popular strategy, where $\nabla\psi$ is approximated by an L^2-projection $\nabla\psi^*$ of $\nabla\hat{\psi}$ is adopted here [1,3]. A superscript asterisk is used in I_j and α_j (i.e. I_j^* and α_j^*) if the exact gradient is replaced by a smoothed version of the computed gradient field.

4. EVALUATION OF THE METHOD

4.1. A stationary problem with singularities

The single-phase porous media flow problem specified in Section 3.1 is considered. There are two particular features of the equation for ψ that calls for adaptivity. First, large pressure gradients appear in the vicinity of the wells, and proper resolution of such steep solutions requires a small mesh size. Second, the permeability K often exhibits very large jumps along geometrically complicated surfaces separating various geological regions. Refinement around wells is addressed in the present section, whereas refinements due to jumps in K are a subject of Section 5.

Consider a 2D domain $\Omega = [-1, 1] \times [-1, 1]$, with an injection well in $x_I = (-0.6, -0.6)$ and an extraction well in $x_E = (0.6, 0.6)$. A simple analytical solution that is convenient to work with, is $\psi = Q \ln(r - r_I) - Q \ln(r - r_E)$, where $r = \|x\|$, $r_I = \|x_I\|$ and $r_E = \|x_E\|$ (note that a norm without subscript implies the Eucledian norm in this paper). This analytical solution is then specified as Dirichlet conditions at the boundary. The velocity is not integrable around the wells, therefore, a small circular area is excluded around each well in the computation of the energy norm. A corresponding 3D problem in $\Omega = [-1, 1]^3$ has also been investigated and gave the same qualitative effects as demonstrated for the comprehensive 2D experiments below.

Figure 5 shows the error, $\log\|\psi - \hat{\psi}\|_{H^1(\Omega)}$, as a function of $\log n$ for six different refinement strategies. The five refined grid types lead to faster convergence rate than a uniform grid. Moreover, the *a priori* geometric criteria perform better than the error indicator I_j^* and a criterion based on the size of $\|\nabla\hat{\psi}\|$. This demonstrates that insight into the problem can devise better refinement strategies than widely used automatic error estimation procedures. The best results in this 2D test problem were obtained by experimenting with an n-irregular grid ($\square$ symbols in Figure 5) with refinements as indicated in Figure 6. This series comprises independent runs with different n values and the use of repeated refinement. That is why the

□ symbols are not lying on a smoothly varying curve. The results demonstrate that *n*-irregular grids, with significant jumps in the element sizes, can give better performance than the more commonly used one-irregular, smoother grids.

4.2. The advection equation

The grid refinement technique described in this paper is applicable to time-dependent phenomena. For illustration, Figure 7 shows the standard problem of pure advection of a Gaussian hill in a rotating velocity field [29], solved by two uniform and one adaptive grid.

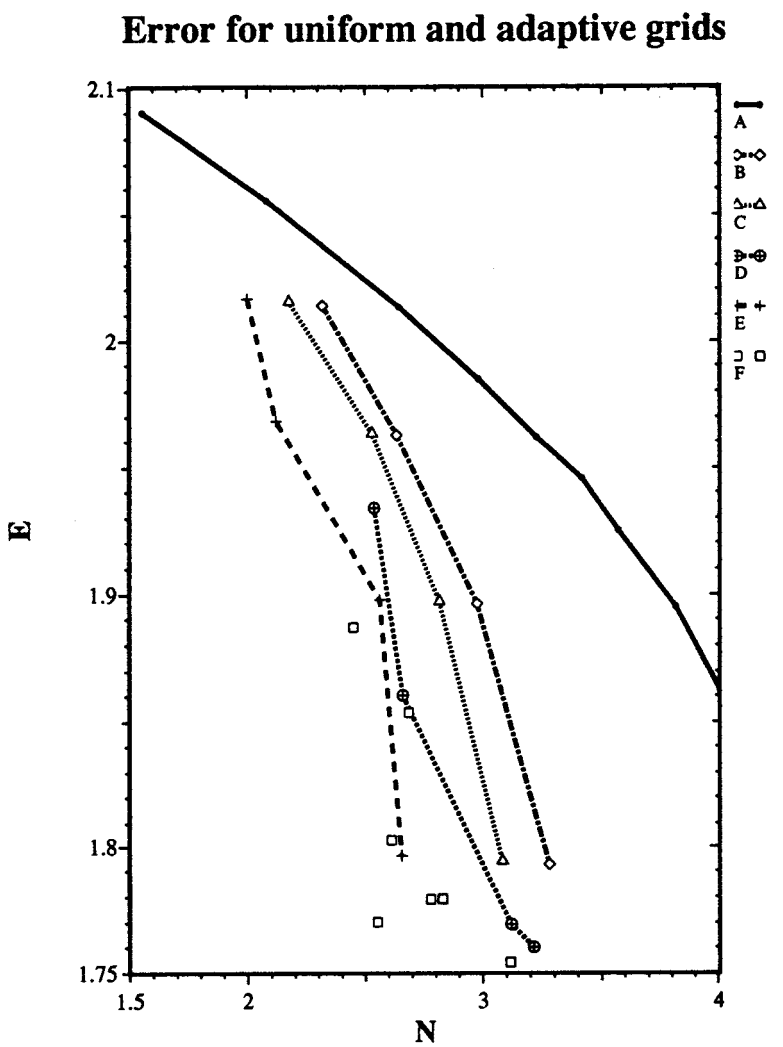

Figure 5. Plot of the error in a Poisson problem with singularities and known analytical solution. The symbol N reflects the logarithm of the number of degrees of freedom, whereas E denotes the logarithm of the $H^1(\Omega)$ norm of the error. Curve A (—), uniform partition; curve B (◇), ZZ-estimator I_j^*, one-irregular grid, bilinear elements; curve C (△), refinement based on the size of $\|\nabla\hat{\psi}\|$, one-irregular grid, bilinear elements; curve D (⊕), refinement inside a prescribed geometric region, one-irregular grid, biquadratic elements; curve E (+), refinement inside a prescribed geometric region, one-irregular grid, bilinear elements; and curve F (□), refinement inside a prescribed geometric region, nested *n*-irregular grid, bilinear elements (see Figure 6).

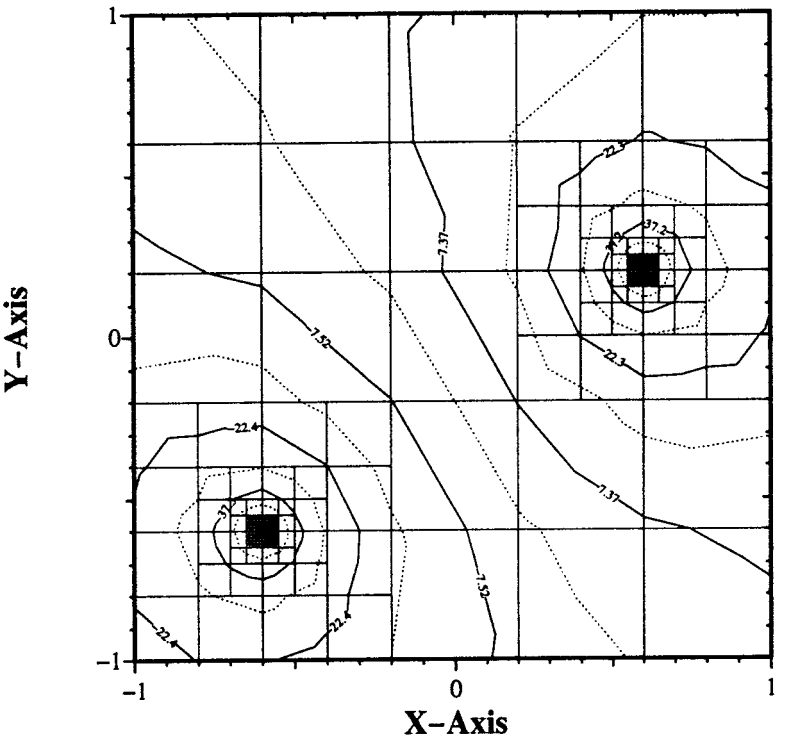

Figure 6. Plot of the grid and the contour lines of the pressure $\hat{\psi}$ in a flow case with two wells. There is a five-irregular grid close to the wells, surrounded by a smoother one-irregular grid.

The numerical method was based on a SUPG formulation [30], backward Euler time scheme and a lumped mass matrix. This approach is far from optimal for the simple advection equation, but often used for more complicated non-linear equations modeling transport in porous media flow. The adaptive mesh gives the same accuracy as a uniform mesh with over four times the number of nodes. Also in this case, n-irregular grids with $n > 1$ are advantageous. The optimal solution in Figure 7 was produced by a geometric refinement criterion, based on measuring the hill width, and a two-irregular grid with three refinement levels. For efficiency, it is preferable to avoid coarsening and instead refine only the underlying coarse grid at selected time levels.

4.3. 3D examples

The refinement method has been applied to problems where Equation (8) is solved on three-dimensional grids. In the case where refinements are performed in the vicinity of wells, the results are similar to those obtained in 2D, and because the 3D series is less comprehensive, without giving new information, only the 2D results in Figure 5 are presented here. However, a free surface groundwater flow model, based on Equation (8) and an advection equation for tracking the water–air interface, has been studied in 3D and will be published elsewhere. Figure 8 illustrates a simple refinement in a 3D box.

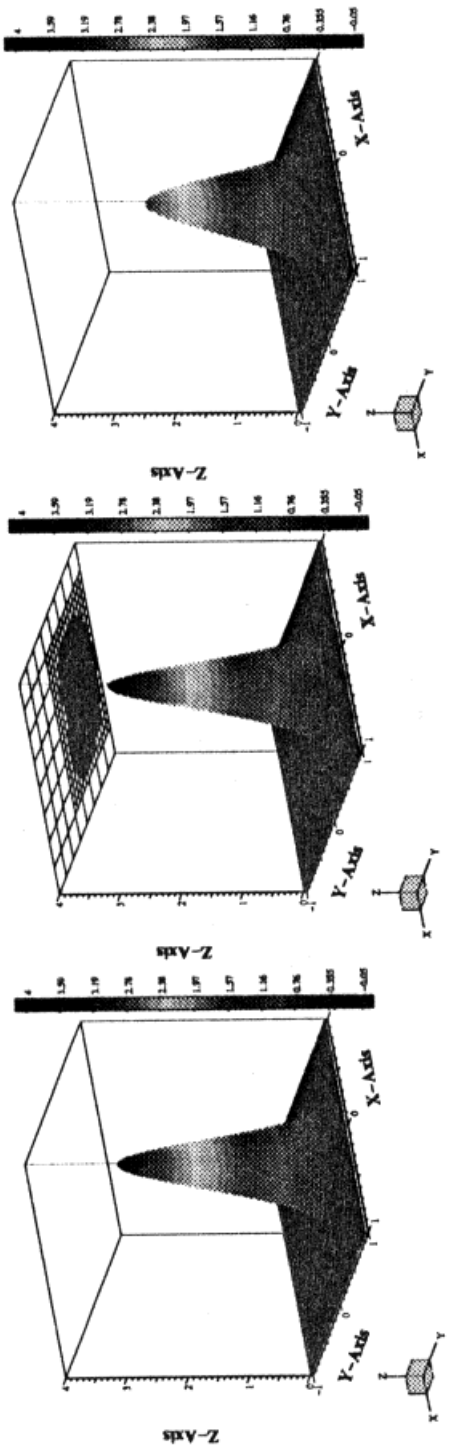

Figure 7. Advection of a Gaussian hill. (a) uniform grid 40401 nodes; (b) adaptive grid with 9286 nodes (on average); and (c) uniform and 10201 nodes.

5. A DOMAIN DECOMPOSITION METHOD

5.1. Basic ideas

In this section a new domain decomposition-type method is presented to improve a coarse mesh finite element field by using the suggested refinement procedure. A well-known classical approach for improving the accuracy of stress computations on a coarse mesh is to pick out small parts of the mesh, apply the coarse grid solution as boundary condition and compute a local solution using a fine grid [18,19]. Modern domain decomposition methods can treat the same problem by iterating over the proper interface conditions [13] or using overlapping domains [31]. Instead of using the coarse mesh solution as boundary conditions for the local fine grid analysis, we propose to extend the local grid with coarse grid elements so that the whole domain is covered in the local analysis. This means that the physically correct boundary conditions are also applied in the local analysis. From a practical point of view, the grid needed in the local analysis is simply a coarse mesh with an n-irregular fine mesh over a local region. This local region could be a single coarse mesh element, but to improve the accuracy, the local mesh can also consist of a coarse mesh element and its neighbors [18,19]. Including the neighbors might reduce the effect of the irregular nodes and the constraints if it is not intended to make use of the solution over the neighbor elements.

Let $G = \{\Omega_1^c, \ldots, \Omega_{n_c}^c\}$ denote the set of coarse grid elements. A refined mesh G_k is based on G, but where Ω_k^c and its adjacent elements are refined into $q \times q$ grids, producing a global $(q-1)$-irregular mesh. One can create $n_r \leq n_c$ such refined meshes G_k, each associated with a coarse grid element. The advantage of working with n_r separate, refined grids, and not a single grid where all the n_r coarse grid elements are refined simultaneously, is only evident when the problem size prevents the single grid approach or the whole computation is to be performed

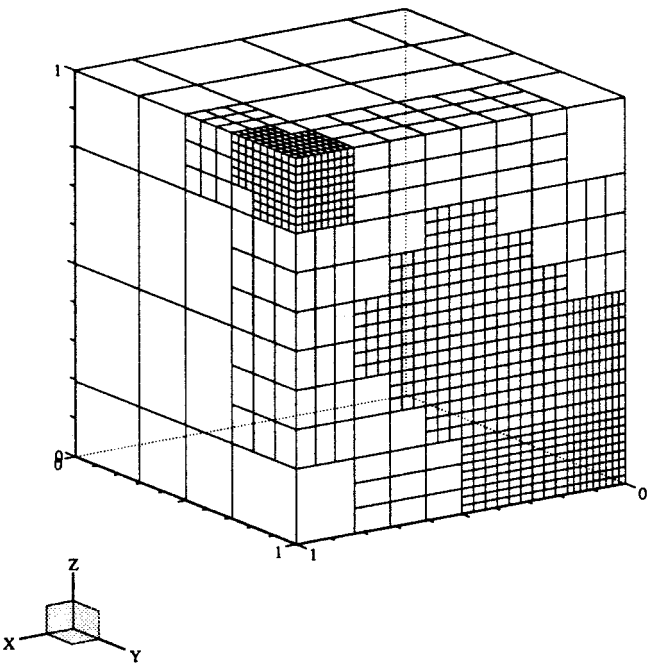

Figure 8. Example of 3D refinements in a box shaped domain.

in parallel. An algorithm is presented below where the n_r grids can be worked in sequence or in parallel.

Having computed the n_r independent solutions $\psi^{(i)}$ over the locally refined grids G_i, $i = 1, \ldots, n_r$, a composite solution is established over a fine grid G_F where all the n_r target elements for refinement are refined. Let $\psi^{(i)}$ denote a function that vanishes at the nodes outside Ω_i^c and that equals $\psi^{(i)}$ at the nodes that belong to the refined Ω_k^c element. Two alternative procedures are suggested.

5.1.1. Method 1. The fine grid solution over G_F is simply established as a superposition of the independent solutions $\psi^{(i,0)}$, $i = 1, \ldots, n_r$. On the interfaces between refined subdomains we can have contributions from different $\psi^{(j,0)}$ fields. The composite field is then taken as the arithmetic mean $n_r^{-1} \Sigma_j \psi^{(j,0)}$. In a practical implementation, only the contributing j values are taken into account. Other alternatives might employ partition of unity functions as weights, such that there is a larger degree of exchange of information between the subdomains, but this is not considered further herein.

5.1.2. Method 2. In this extended method an improved coarse grid solution is computed by adding information from the independent solutions $\psi^{(i,0)}$. Thereafter, the improved coarse grid solution and the solutions $\psi^{(i,0)}$ are combined. The improved coarse grid computation employs the following expression for $\hat{\psi}$ over Ω_k^c

$$\hat{\psi}(x) = \hat{\psi}^{(k,0)}(x) + \sum_j (N_j(x)\hat{\psi}_j - N_j(x)\hat{\psi}_j^{(k,0)}), \quad x \in \Omega_k^c, \tag{11}$$

where $\hat{\psi}^{(k,0)}$ has a standard finite element expansion over the fine grid, N_j denotes coarse grid basis functions, $\hat{\psi}_j^{(k,0)}$ is the value of $\hat{\psi}^{(k,0)}$ at the coarse mesh nodes in Ω_k^c, and $\hat{\psi}_j$ denotes the values of $\hat{\psi}$ at the coarse mesh nodes. Inserting this in the model problem results in a modified coarse grid problem. The following contribution is obtained from element Ω_k^c

$$\sum_j a(N_i, N_j)\hat{\psi}_j = L(N_i) + \sum_j a(N_i, N_j)\hat{\psi}_j^{(k,0)} - a(N_i, \hat{\psi}^{(k,0)}).$$

The term $a(N_i, \hat{\psi}^{(k,0)})$ must be integrated carefully because $\hat{\psi}^{(k,0)}$ is defined on a finer scale than $\hat{\psi}$. Obviously, the ordinary Gauss–Legendre quadrature can be used over the fine grid elements. Finally, the solution is computed over the fine grid G_F by a superposition of the contributions according to Equation (11) in the same manner as in method 1.

In the case where only Ω_k^c and not its neighbors are refined in a G_k grid, methods 1 and 2 are denoted by 1s and 2s (single element refinement). This domain decomposition procedure can be very elegantly implemented utilizing OOP, where the simulator itself is an object. Details are given elsewhere [32].

5.2. Numerical examples

To illustrate the proposed method, numerical experiments have been conducted in two examples where the pressure equation was solved

$$-\nabla \cdot \left[\frac{k}{\mu} \nabla \psi \right] = f \text{ on } [0, 1] \times [0, 1].$$

In the first test example, K is chosen as a smooth function, either constant, or

$$K = 1 + 0.95 \sin \frac{2\pi}{\lambda} x \sin \frac{2\pi}{\lambda} y, \tag{12}$$

Table I. Solution error for a test problem with analytical solution and $K = 1$

q	Method 1	Uniform mesh	Method 2	Method 2s
2	3.76e-04	3.55e-04	3.64e-04	3.65e-04
4	3.67e-05	2.23e-05	2.92e-05	8.83e-05
8	5.52e-06	1.40e-06	3.79e-06	1.94e-05
16	1.14e-06	8.75e-08	7.29e-07	4.67e-06

The coarse mesh is a uniform 10×10 grid ($n_c = 100$).

Table II. Solution error for a test problem with analytical solution and $K = 1$

q	Method 1	Uniform mesh	Method 2	Method 2s
2	6.11e-03	5.53e-03	5.46e-03	6.06e-03
4	1.29e-03	3.55e-04	3.60e-04	1.27e-03
8	2.74e-04	2.23e-05	2.68e-05	2.71e-04
16	6.54e-05	1.40e-06	2.71e-06	6.46e-05

The coarse mesh is a uniform 5×5 grid ($n_c = 25$).

where $2\pi/\lambda$ should not equal an integer. The right hand side f and the boundary conditions are adjusted such that $\psi = \cos \pi x \cos \pi y$, is the solution of the problem. It is then easy to compute the true approximation error in the numerical method. Tables I and II show how the error varies with the methods and the value of the subdivision parameter q in the case $K = 1$. The uniform mesh corresponds to $q \times q$ refinement of all the coarse mesh elements. We see from Table III that the differences between the methods are fairly small when the coarse grid has 10×10 elements. Making the coarse grid even coarser (5×5 elements) leads to varying performance of the methods. Table IV shows that the simple method 1 increases the error by approximately an order of magnitude compared with a uniform fine mesh computation. Method 2 is capable of coming very close to the uniform fine mesh computations, while refining only single elements (method 2s) exhibits the same behaviour as method 1. When K is given by Equation (12) more encouraging results are achieved, as shown in Tables III and IV. For this more challenging example, the error introduced by various approximation methods is much smaller. The conclusion is that the suggested overlapping domain decomposition procedure is a valuable tool for solving at least these types of elliptic problems with smooth solution.

The second test example has $f = 0$ and piecewise constant K with a large jump. Inside the ellipse, $K = 10^{-8}$, and outside $K = 1$. This problem models porous media flow around an almost impermeable elliptical-shaped, geological obstacle. If the discontinuities of K do not coincide with the element surfaces, significant inaccuracies in the computations can occur. Ideally, sophisticated gridding techniques would be applied to let the element surfaces follow the geological subdomains. From a practical point of view it would, however, be very advantageous to have a method that also works satisfactorily in the case where the boundaries of the geological subdomains intersect with the finite element mesh. Decreasing the mesh size in the vicinity of such intersections represents a mean for increasing the accuracy. Typical G_k grids are depicted in Figure 9.

Figure 10 compares velocities $(K/\mu)\nabla\psi$ computed by method 2, with velocities from a fine grid uniform mesh computation. When using the underlying coarse mesh only, there is almost no effect of the low-permeable region on the velocity field. One should recognize that this is

Table III. Solution error for a test problem with analytical solution and smoothly varying K according to Equation (12) with $\lambda = 0.17$

q	Method 1	Uniform mesh	Method 2	Method 2s
2	3.1553e-03	2.6094e-03	2.7943e-03	2.8390e-03
4	6.4103e-04	5.5724e-04	5.7678e-04	6.1735e-04
8	3.4151e-04	3.1021e-04	3.1848e-04	3.1023e-04
16	2.0173e-05	1.7318e-05	1.8391e-05	2.3523e-05

The coarse mesh is a uniform 10×10 grid ($n_c = 100$).

Table IV. Solution error for a test problem with analytical solution and smoothly varying K according to Equation (12) with $\lambda = 0.17$

q	Method 1	Uniform mesh	Method 2	Method 2s
2	1.9326e-02	1.9341e-02	1.9399e-02	1.9483e-02
4	2.6580e-03	2.6094e-03	2.6243e-03	3.8218e-03
8	5.6916e-04	5.5724e-04	5.6381e-04	8.2821e-04
16	3.1843e-04	3.1021e-04	3.1287e-04	3.8427e-04

The coarse mesh is a uniform 5×5 grid ($n_c = 25$).

a very challenging problem. The effect of an obstacle is only detected in a small region at a time, and the quantity presented in the figures is the *derivative* of the primary unknown.

6. CONCLUSION

A unified finite element mesh refinement algorithm that works in 1D, 2D, 3D and in even higher dimensions has been presented and investigated in this paper. The algorithm can handle

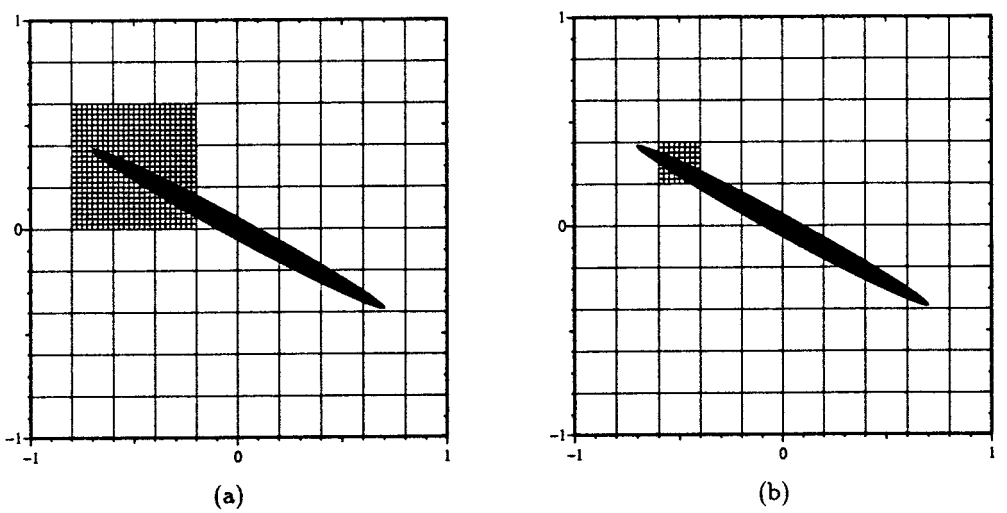

Figure 9. Sketch of a G_k grid as used in the domain decomposition-type methods in Section 5, with refinements in the vicinity of a low-permeable formation, illustrated by the black ellipse. (a) G_k has refinement of coarse grid element Ω_k^c and its neighbors; (b) G_k has refinement inside Ω_k^c only.

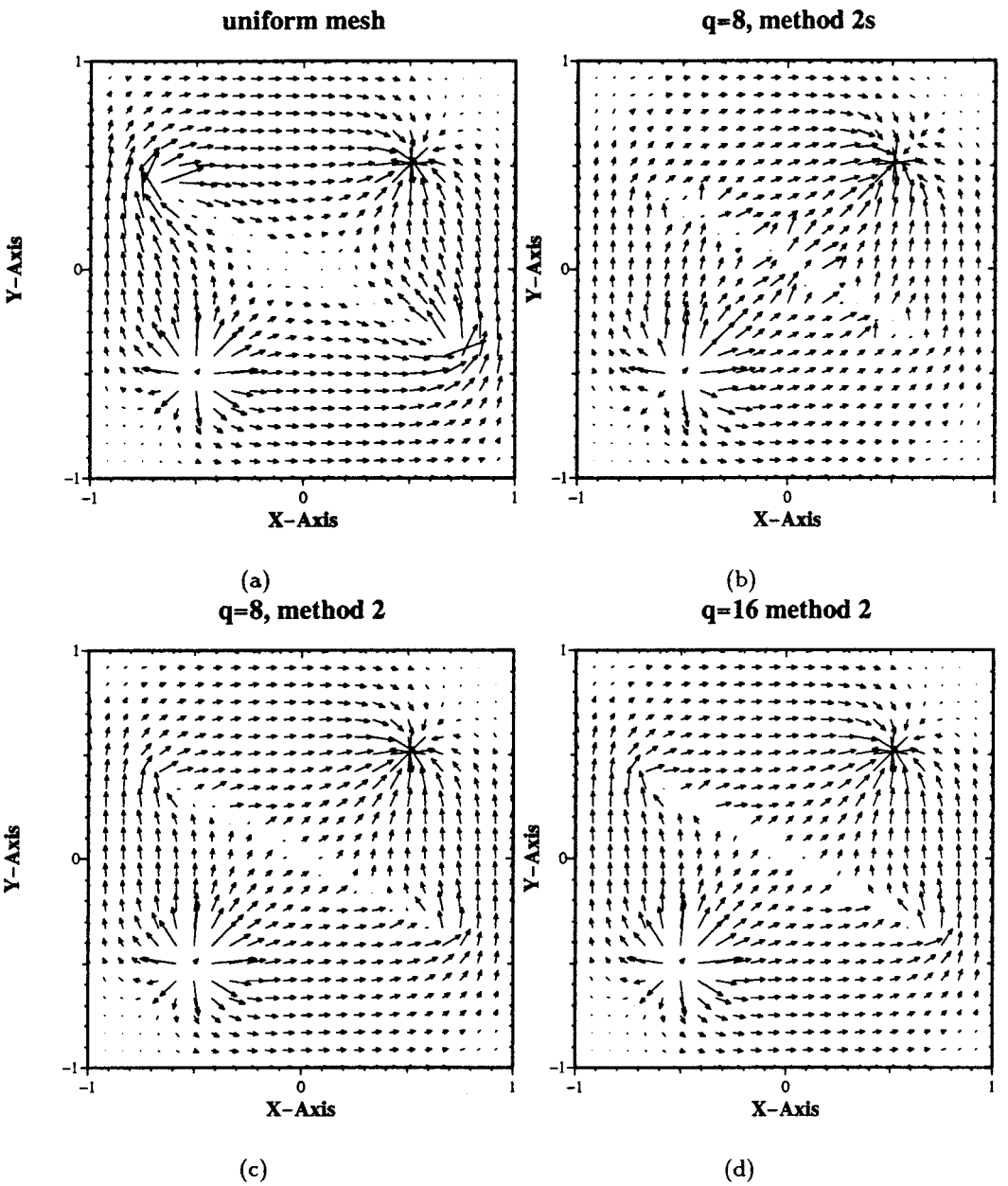

Figure 10. Velocity fields $(K/\mu)\nabla\psi$ in a flow problem with a low-permeable region ($K = 1$ outside the obstacle and $K = 10^{-8}$ inside). (a) uniform find grid solution, with 80×80 bilinear elements; (b) approximate numerical solution, with method 2s, 10×10 coarse mesh, and $q = 8$; (c) approximate numerical solution, with method 2, 10×10 coarse mesh, and $q = 8$; (d) approximate numerical solution, with method 2, 10×10 coarse mesh, and $q = 16$.

tensor product Lagrange elements of arbitrary polynomial degree, but is currently limited to h-refinement. The subdivision of elements preserves the element type, i.e. no new transition elements are introduced. Irregular nodes appear as a result of the refinement process, and lead to additional constraint equations. It has been shown that in addition to the common

continuity requirement of the primary unknown field at irregular nodes, continuity of the flux is also important, and this additional constraint can be incorporated by a symmetrization of the original constraint equation. The resulting meshes can be n-irregular, with the common choice $n = 1$ as a special case. By numerical examples related to single-phase porous media flow we have demonstrated that n-irregular meshes can be advantageous for $n > 1$. We have also experimented with various refinement criteria without reaching any unique conclusions. Criteria based on geometric information about the domain, the coefficients and the solution can sometimes be clearly superior to criteria based on standard and widely used error estimation procedures. This has been illustrated in a porous media flow problem with singularities at the wells.

The n-irregular refinements for $n > 1$ lead naturally to locally refined grids that are extended by coarse grid elements to cover the whole physical domain. Such grids have been used in a domain decomposition-type approach and the performance of various strategies demonstrated.

A major advantage of the present refinement algorithm and its object-oriented implementation is that existing (Diffpack) finite element simulators can immediately take advantage of adaptive grids by incorporating about ten extra lines of code in the managing part of the program. Since there is no need to modify thoroughly debugged numerics, utilization of adaptive grids in a simulator is a simple, reliable and efficient process. Simulators have already been equipped for injection moulding and free surface groundwater flow with adaptivity, and results for these cases will be reported elsewhere.

ACKNOWLEDGMENTS

The authors thank Halvard Fjær for valuable discussions of the material in Appendix A.

APPENDIX A. FLUX CONTINUITY REQUIREMENTS

The incorporation of the constraint equation (3) can be performed in two ways. Method N consists of a direct insertion of Equation (3) into the linear system (2) which is assembled without paying any attention to the constraint. This yields Equation (4) and appears to be a common strategy in some of the fundamental literature [33,34]. Besides introducing a non-symmetric stiffness matrix even if the matrix A_{ij} in Equation (2) is symmetric, this procedure does not preserve flux continuity in a weak sense. We will show in detail how a physically sound handling of the fluxes is equivalent to a symmetry preserving introduction of the constraint equation (3) in (2). This results in Equation (5), and this approach is referred to as method S in the following. Note that method S can be viewed as applying method N followed by a row modification of the system. The row modification uses information from the assembled finite element equation for the constrained node. This information is unused in method N.

First the class of problems to be considered is defined, and a definition of the flux is given. Consider a term on the form $\nabla \cdot u$ in a partial differential equation. It is assumed that u is a vector (as in heat transport or porous media flow) or a tensor (as in elasticity). The associated flux through a surface Γ is then $\int_\Gamma u \cdot n$. In the finite element method, $\nabla \cdot u$ is usually integrated by parts, and the flux term associated with one element is annihilated by the flux term of a neighbor element at all internal boundaries. This result is based on the assumption of a continuous flux. In many problems flux continuity follows directly from physical considerations.

Consider an internal boundary Γ between finite elements. At one side of Γ there are coarse grid elements, while there are fine grid elements at the other side (see Figure 11). On this interface, there is a set J of irregular, constrained nodes. They are constrained by a set I of parent nodes. The total set of nodes on Γ is $K = I \cup J$. The finite element equation with parent node A is investigated in Figure 11. The node A contributes to the constraint equations of all the irregular nodes in J. The figure is in 2D, but the following considerations are also valid for higher dimensions, for any interpolation order, and for any element subdivision. All quantities restricted to coarse elements have a prime, and quantities without primes refer to fine grid elements.

Flux continuity in the weak sense implies

$$\int_\Gamma N_A u \cdot n \, d\Gamma + \int_\Gamma N'_A (u \cdot n)' \, d\Gamma = 0. \tag{13}$$

Without refinement, $N_A = N'_A$ along Γ, and since we analytically assume $u \cdot n = -(u \cdot n)'$, the integrals sum to zero. However, consider the situation in Figure 11, and the contribution b_A to the right-hand-side of the discrete equation system

$$b_A = \sum_e \int_{\Gamma_e} N_A u \cdot n \, d\Gamma - \int_\Gamma N'_A u \cdot n \, d\Gamma. \tag{14}$$

Here, Γ_e is the part of Γ restricted to fine grid element e. This right-hand-side term will in general not vanish. The incorporation of the constraint equation (2) by method N does not affect this result. Now, it will be shown that the symmetric implementation of the constraint equation in method S modifies the right-hand-side contribution (14) and, in fact, totally removes it and thereby re-establishes the flux compatibility.

When the constraint equation is written in terms of the basis functions as in Equation (1), the row modification produces a contribution from each constrained node j on Γ to the right-hand-side of the equation of node A reading

$$C_A b_j = N'_A(\xi_j) b_j = N'_A(\xi_j) \sum_e \int_{\Gamma_e} N_j u \cdot n \, d\Gamma_e. \tag{15}$$

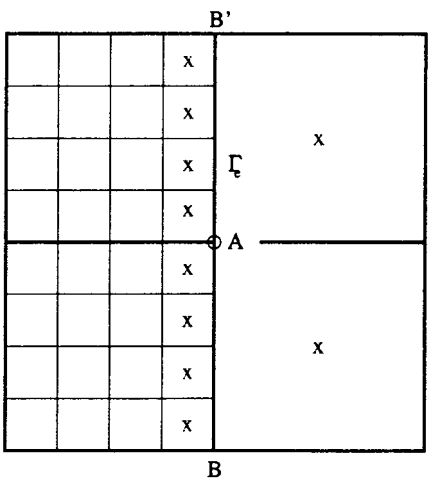

Figure 11. Sketch of a refined grid and where fluxes are to be computed along the interface Γ, defined by the line BB′ in the figure. All elements containing a cross contribute to the flux integrals.

Now, the right-hand-side of equation A after the row modification reads

$$b_A = \sum_e \int_{\Gamma_e} N_A u \cdot n \, d\Gamma_e - \int N'_A u \cdot n \, d\Gamma + \sum_{j \in J} \sum_e N'_A(\xi_j) \int_{\Gamma_e} N_j u \cdot n \, d\Gamma_e. \tag{16}$$

By the simple property of the basis functions

$$N'_A(\xi_j) = \begin{cases} 1 & \text{if } j \text{ is a parent node on } \Gamma \text{ and equal to } A \\ 0 & \text{if } j \text{ is a parent node on } \Gamma \text{ and equal to } A' \end{cases} \tag{17}$$

The following term can be rewritten

$$\sum_e \int_{\Gamma_e} N_A u \cdot n \, d\Gamma_e = \sum_{i \in I} \sum_e N'_A(\xi_i) \cdot \int_{\Gamma_e} N_i u \cdot n \, d\Gamma_e. \tag{18}$$

The two sum terms in Equation (16) can then be merged

$$b_A = -\int N'_A u \cdot n \, d\Gamma + \sum_{k \in K} \sum_e N'_A(\xi_k) \int_{\Gamma_e} N_k u \cdot n \, d\Gamma_e. \tag{19}$$

Providing tensor product elements of the Lagrange type are used and are of the same order on each side, then for each subarea Γ_e a simple connection can be formed between the basis functions in the refined part and in the coarse part

$$N'_A = \sum_{k \in \Gamma_e} N'_A(\xi_k) \cdot N_k, \tag{20}$$

where the sum is taken over the nodes on the interface belonging to element e. Splitting the integral over Γ in Equation (19) into a sum over the subintervals Γ_e and introducing the relation (20) gives

$$b_A = -\sum_e \sum_{k \in \Gamma_e} N'_A(\xi_k) \int_{\Gamma_e} N_k u \cdot n \, d\Gamma_e + \sum_{k \in K} \sum_e N'_A(\xi_k) \int_{\Gamma_e} N_k u \cdot n \, d\Gamma_e. \tag{21}$$

Realizing that the sums can be interchanged, the result is the desired one: $b_A = 0$.

It has been shown that the symmetric incorporation of the constraint equations has ensured that the flux terms for a general parent node A across the internal irregular element interface Γ are zero when it is assumed that the physical flux is continuous, and tensor product elements of the Lagrange type are used.

REFERENCES

1. O.C. Zienkiewicz and J. Z. Zhu, 'A simple error estimator and adaptive procedure for practical engineering analysis', *Int. j. numer. methods eng.*, **24**, 337–357 (1987).
2. T. Strouboulis and K.A. Haque, 'Recent experiences with error estimation and adaptivity, part ii: Error estimation for *h*-adaptive approximations on grids of triangles and quadrilaterals', *Comput. Methods Appl. Mech. Eng.*, **100**, 359–430 (1992).
3. R.W. Lewis, H.C. Huang, A.S. Usmani and J.T. Cross, 'Finite element analysis of heat transfer and flow problems using adaptive remeshing including application to solidification problems', *Int. j. numer. methods eng.*, **32**, 767–781 (1991).
4. R. Löhner, 'An adaptive finite element scheme for transient problems in CFD', *Comput. Methods Appl. Mech. Eng.*, **61**, 323–338 (1987).
5. R. Beck, B. Erdman and R. Roitzsch, 'An object-oriented adaptive finite element code: Design issues and applications in hyperthermia treatment planning', in E. Arge, A.M. Bruaset and H.P. Langtangen (eds.), *Modern Software Tools for Scientific Computing*, Birkhäuser, Basel, 1997.
6. J.A. Trangenstein, 'Adaptive mesh refinement for wave propagation in nonlinear solids', *SIAM J. Sci. Comput.*, **16**, 819–839 (1995).

7. L. Demkowicz, J.T. Oden, W. Rachowicz and O. Hardy, 'Toward a universal h–p adaptive finite element strategy, part I. Constrained approximation and data structure', *Comput. Methods Appl. Mech. Eng.*, **77**, 79–112 (1989).
8. K.C. Chellamuthu and N. Ida, 'Algorithms and data structures for 2d and 3d adaptive finite element mesh refinements', *Finite Elements in Analysis and Design*, **17**, 205–229 (1994).
9. D.J. Morton, J.M. Tyler and J.R. Dorroh, 'A new 3d finite element for adaptive h-refinement in 1-irregular meshes', *Int. j. numer. methods eng.*, **38**, 3989–4008 (1995).
10. A.K. Gupta, 'A finite element for transition from a fine to a coarse grid', *Int. j. numer. methods eng.*, **12**, 35–45 (1978).
11. M.J. Berger, 'Data structures for adaptive mesh refinement', in I. Babuska, J. Chandra and J.E. Flaherty (eds.), *Adaptive Computational Methods for Partial Differential Equations*, SIAM, 1983.
12. M.J. Berger and J. Oliger, 'Adaptive mesh refinement for hyperbolic partial differential equations', *J. Comput. Phys.*, **53**, 484–512 (1984).
13. J.H. Bramble, R.E. Ewing, J.E. Pasciak and A.H. Schatz', 'A preconditioning technique for the efficient solution of problems with local grid refinement', *Comput. Methods Appl. Mech. Eng.*, **67**, 149–159 (1988).
14. O. Sævareid, 'On local grid refinement techniques for reservoir flow problems', *Ph.D. Thesis*, Department of Mathematics, 1990.
15. K.C. Wang and G.F. Carey, 'Adaptive grids for coupled viscous flow and transport', *Comput. Methods Appl. Mech. Eng.*, **82**, 365–383 (1990).
16. M. Lemke, K. Witsch and D. Quinlan, 'An object-oriented approach for parallel self adaptive mesh refinement on block structured grids,' in W. Hackbusch and G. Wittum (eds.), *Adaptive Methods—Algorithms, Theory and Applications, 1993.* Proc. of the 9th GAMM-Seminar, Kiel, January 22–24, 1993.
17. J.-L. Liu, I.-J. Lin, M.-Z. Shih, R.-C. Chen and M.-C. Hsieh, 'Object-oriented programming of adaptive finite element and finite volume methods,' *Appl. Numer. Math.*, **21**, 439–467 (1996).
18. K.M. Mao and C.T. Sun, 'A refined global-local finite element analysis method,' *Int. j. numer. methods eng.*, **32**, 29–43 (1991).
19. I. Hirai, Y. Uchiyama, Y. Mizuta and W.D. Pilkey, 'An exact zooming method', *Finite Element Analysis and Design*, **1**, 61–69 (1985).
20. R.E. Ewing, 'Adaptive mesh refinements in large-scale fluid flow simulation', in I. Babuska, O.C. Zienkiewicz, J. Gago, and E.R. de A. Oliveira (eds.), *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, Wiley, New York, 1986, pp. 299–314.
21. E. Arge, A.M. Bruaset, P.B. Calvin, J.F. Kanney, H.P. Langtangen and C.T. Miller, 'On the efficiency of C++ for scientific computing', in M. Dachlen and A. Tveito (eds.), *Mathematical Models and Software Tools in Industrial Mathematics*, Birkhäuser, Basel, 1997.
22. E. Arge, A.M. Bruaset and H.P. Langtangen, 'Object-oriented numerics', in M. Dæhlen and A. Tveito (eds.), *Mathematical Models and Software Tools in Industrial Mathematics*, Birkhäuser, Basel, 1997.
23. Diffpack world wide web home page, http://www.oslo.sintef.no/diffpack/.
24. A.M. Bruaset and H.P. Langtangen, 'Tools for solving partial differential equations; Diffpack', in M. Dæhlen and A. Tveito (eds.), *Mathematical Models and Software Tools in Industrial Mathematics*, Birkhäuser, Basel, 1997.
25. J.J. Barton and L.R. Nackman. *Scientific and Engineering C++—An Introduction with Advanced Techniques and Examples*, Addison-Wesley, Reading, MA, 1994.
26. I. Babuska, T. Strouboulis and C.S. Upadhyay, 'A model study of the quality of a posteriori error estimators for linear elliptic problems', *Comput. Methods Appl. Mech. Eng.*, **114**, 307–378 (1994).
27. O.C. Zienkiewicz and J.Z. Zhu, 'The super convergent patch recovery and a posteriori error estimates. part 1: The recovery technique', *Int. j. numer. methods eng.*, **33**, 1331–1364 (1992).
28. O.C. Zienkiewicz and J.Z. Zhu, 'The super convergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity', *Int. j. numer. methods eng.*, **33**, 1365–1382 (1992).
29. P. Devloo, J.T. Oden and T. Strouboulis, 'Implementation of an adaptive refinement technique for the SUPG algorithm', *Comput. Methods Appl. Mech. Eng.*, **61**, 339–358 (1987).
30. A.N. Brooks and T.J.R. Hughes, 'Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equation', *Comput. Methods Appl. Mech. Eng.*, **60**, 199–259 (1982).
31. B. Smith, P. Bjørstad and W. Gropp, *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, 1996.
32. A.M. Bruaset, E. Holm and H.P. Langtangen, 'Increasing the efficiency and reliability of software development for systems of PDEs', in E. Arge, A.M. Bruaset and H.P. Langtangen (eds.), *Modern Software Tools for Scientific Computing*, Birkhäuser, Basel, 1997.
33. G.F. Carey and J.T. Oden. *Finite Elements: Computational Aspects, volume III in the Texas Finite Element Series, first edition*, Prentice-Hall, Englewood Cliffs, NJ, 1986.
34. B.A. Finlayson, *Numerical Methods for Problems with Moving Fronts*, 1st edn., Ravenna Park Publishing, Seattle, USA, 1992.

