COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING 67 (1988) 149–159
NORTH-HOLLAND

A PRECONDITIONING TECHNIQUE FOR THE EFFICIENT SOLUTION OF PROBLEMS WITH LOCAL GRID REFINEMENT*

James H. BRAMBLE
Cornell University, Ithaca, NY 14853, U.S.A.

Richard E. EWING
Department of Mathematics, University of Wyoming, Laramie, WY 82071, U.S.A.

Joseph E. PASCIAK
Brookhaven National Laboratory, Upton, NY 11973, U.S.A.

Alfred H. SCHATZ
Cornell University, Ithaca, NY 14853, U.S.A.

Received 23 April 1986
Revised manuscript received 20 April 1987

We develop a new preconditioning method for elliptic problems which allows for dynamic local grid refinement. The majority of the computation in the implementation of our method involves solution procedures on mesh domains with regular geometry. Accordingly, the resulting algorithms can be effectively vectorized. It seems feasible to incorporate these ideas into existing large-scale simulators without a complete redesign of the simulator.

1. Introduction

The objective of reservoir simulation is to understand the complex chemical, physical, and fluid flow processes occurring in a petroleum reservoir sufficiently well to be able to optimize the recovery of hydrocarbon. Many of the phenomena which govern recovery processes have an extremely local character and must be resolved for successful modeling. Although the techniques presented here are sufficiently general to apply to many different large-scale applications, in this paper we concentrate on a representative problem arising in petroleum reservoir simulation. We consider the problem of modeling the local flow behavior around wells.

The major input and output from a reservoir in various production procedures is through wells and hence it is important to obtain an accurate approximation to the flow nearby. If fluid

* This manuscript has been authored under contract number DE-AC02-76CH00016 with the U.S. Department of Energy. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. This work was also supported in part under the National Science Foundation Grants Nos. DMS84-05352 and DMS85-04360, under U.S. Army Research Office Contract No. DAAG29-84-K-0002, and under the Air Force Office of Scientific Research, Contract Nos. ISSA86-0026 and AFOSR-85-0017.

flow rates are specified at injection or production wells, the use of Dirac delta functions as point sources and sinks in the mathematical equations has been shown to be a good model for well-flow behaviour beyond some minimal distance away from the wells. In this case, the pressure (which determines the flow) grows like $\ln r$, where r is the distance to that well. Another well model involves specification of a bottom hole pressure as a boundary condition. This well model also gives rise to a logarithmic growth in pressure up to a finite specified pressure. Because of the logarithmic behavior, accurate pressure approximations require local grid refinement.

As an example of the governing equations for the pressure p, of a fluid in a horizontal reservoir $\Omega \subset \mathbb{R}^2$, we take

$$-\nabla \cdot \frac{k}{\mu} \nabla p = f \quad \text{in } \Omega . \tag{1.1}$$

Here, μ is the viscosity and k is the permeability. We take no flow boundary conditions, i.e. we assume

$$\frac{k}{\mu} \frac{\partial p}{\partial \nu} = 0 \quad \text{on } \partial\Omega , \tag{1.2}$$

where $\partial/\partial\nu$ is in the direction of the outward normal vector on $\partial\Omega$. For the existence of p we assume that the mean value of f is zero and for uniqueness we impose that p have mean value zero. If fluid flow rates at injection and production wells are specified via Dirac delta functions at the N_w wells x_i with associated flow rates q_i, then

$$f = \sum_{i=1}^{N_w} \delta(x - x_i) q_i . \tag{1.3}$$

Several techniques which assume radial flow near the well have been used to obtain local properties of p. One such technique involves subtracting out the singular behavior of p around the wells [10, 11, 14]. A radial flow assumption is probably not bad around injection wells but is inadequate for production wells. For production wells, different techniques, such as local grid refinement, are often needed. It has been shown [22, 23] that appropriate local grid refinement around these singularities can greatly increase the accuracy throughout the reservoir.

In an involved production strategy, new wells are drilled and old wells are often shut down to produce better sweep efficiency by the injected fluid and increase the hydrocarbon recovery. Thus the need to dynamically turn wells on or off necessitates the ability to add or remove local refinements around the wells of the simulation without regenerating the entire grid. Many of the current solution algorithms are highly vectorized, depend upon a regular matrix structure, and do not allow the dynamic changing of the number of grid points or elements. The ability to have truly local refinement and 'derefinement' capabilities often necessitates the use of a fairly complex data structure. Several data structures with these properties have been described in the literature (e.g. [3, 5, 13, 15, 21]). Such specialized data structures and corresponding linear solution algorithms will have a large overhead in field-scale applications due to element insertion and deletion [3, 5, 15, 21]. The aim of this paper is

to introduce new techniques which will greatly reduce overhead requirements, allow dynamic local grid refinement, and can be incorporated in existing codes.

We will develop a fast solution method for the approximation of problems requiring mesh refinement. This technique is related to various domain decomposition methods [cf. 8, 9]. High accuracy throughout the computational region is obtained by incorporating local refinements around wells. A composite grid is obtained by superimposing these refinements on a quasi uniform grid on the original domain. The model equation (1.1) is then discretized on this composite grid via finite element techniques. A new domain decomposition variant is developed to efficiently solve the resulting matrix equations. This involves the development of a preconditioner. This preconditioner is novel in that the task of computing its inverse applied to a vector reduces to the solving of separate matrix systems for the local refinements and the matrix system for the quasi-uniform grid on the original domain. Note that this quasi-uniform grid overlaps the regions of local refinement and its corresponding matrix problem remains invariant when local refinements are dynamically added or removed. This local refinement technique can be incorporated in existing reservoir codes without extensive modification. Furthermore, if the nodes on the quasi-uniform grid are chosen in a regular pattern, highly vectorizable algorithms for the solution of the corresponding matrix system can be developed.

Extensions of earlier domain decomposition algorithms [6–8, 18] to situations involving local refinements are possible. However, these extensions involve the solution of coarse-grid problems with the regions corresponding to the refinements removed. Thus the coarse-grid problem would change as refinements are added or removed. More complex solution procedures would have to be used to handle this problem since many of the original grid nodes would no longer be in the resulting problem.

The outline of the remainder of the paper is as follows. We describe the preconditioning technique in Section 2. In Section 3, we give numerical examples which illustrate both the rate of convergence for the iterative algorithm as well as the effectiveness of refinement on problems with logarithmic-type singularities.

2. The preconditioner and its implementation

As stated in the introduction, we shall develop a preconditioner for the discrete problem which results from finite element approximation with composite grids. In reservoir applications, a uniform grid is chosen which can resolve the large-scale features of the reservoir problem and which leads to a problem of manageable size for solution. This can be thought of as the grid chosen by a standard existing reservoir simulation code. Next, areas where local refinement is desired are identified via some adaptive refinement criteria [1–5, 12, 13, 15–17, 19, 21]. A composite grid is formed by superimposing the refinement and uniform grids. This is best illustrated by an example.

The example described will be used to study the effect of mesh refinements applied to a problem on a square whose solution exhibits singular behavior at the lower left- and upper right-hand corners. We start with the regular triangulation of the square illustrated in Fig. 1. The composite grid is formed by first removing two subsquares around the singular points (see Fig. 2) and then replacing them by either the radial like refinement of Fig. 3 or the regular refinement of Fig. 4. Figure 5 gives a composite grid where the lower (resp. upper) subsquares

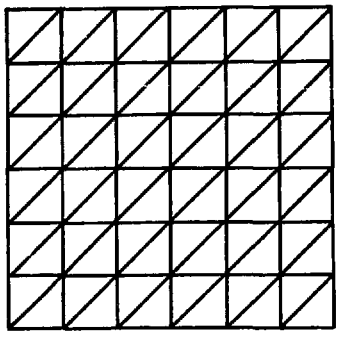

Fig. 1. The original coarse grid on Ω.

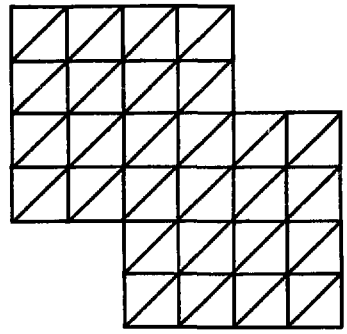

Fig. 2. The grid with subsquares removed.

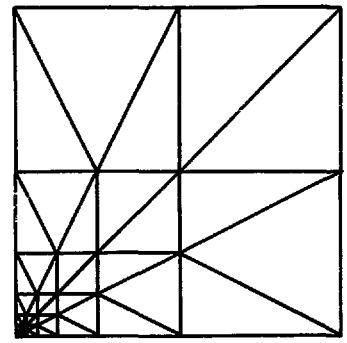

Fig. 3. A 'radial-like' refinement subgrid.

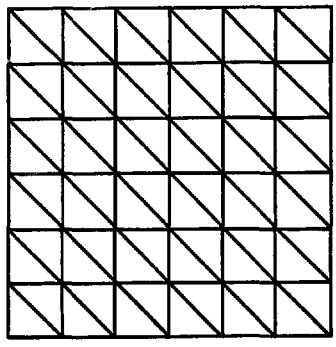

Fig. 4. A regular refined subgrid.

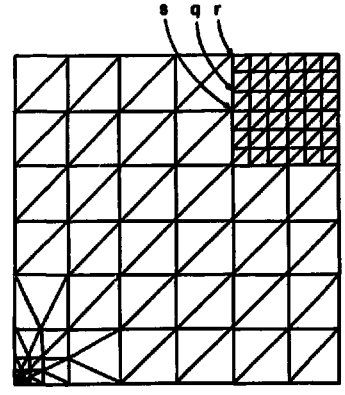

Fig. 5. A composite grid.

were replaced by the refinements of Fig. 3 (respectively Fig. 4). Let Ω_2 be the union of the refinement subregions and $\Omega_1 \equiv \Omega/\Omega_2$.

We next define subspaces of approximating functions on the composite grid. For simplicity of presentation, we shall only consider the most standard approximation subspaces even though many generalizations are possible. Let S_h be the collection of functions which are linear when restricted to the triangles of the composite mesh and continuous on the domain. We note that with such definitions the nodes like q in Fig. 5 can be considered to be 'slave' nodes in the sense that continuity implies that the value of a function in S_h at q is the linear interpolant of the values of the function at r and s. We also note that the radial-like refinement does not generate any slave nodes. S_h^c will denote the subspace of continuous and piecewise linear functions on the coarse grid. This grid is exemplified by Fig. 1. The subspace of S_h of functions with mean value zero will be denoted by $\tilde{S}_h$.

Multiplying (1.1) by an arbitrary (sufficiently regular) function φ, integrating by parts, and using (1.2), we see that the solution p satisfies

$$A(p, \varphi) = (f, \varphi), \tag{2.1}$$

where

$$A(u, v) = \int_\Omega \frac{k}{\mu} \nabla u \cdot \nabla v \, dx,$$

and

$$(u, v) = \int_\Omega uv \, dx .$$

The Galerkin approximation to (2.1) is: Find $P \in \tilde{S}_h$ satisfying

$$A(P, \theta) = (f, \theta) \quad \text{for all } \theta \in \tilde{S}_h . \tag{2.2}$$

As indicated earlier we shall calculate P by a preconditioned iterative method. Thus (see [8, 9]) the problem is equivalent to defining another symmetric positive-definite form B satisfying two criteria. First, given g, the problem of finding $W \in \tilde{S}_h$ satisfying

$$B(W, \theta) = (g, \theta) \quad \text{for all } \theta \in \tilde{S}_h \tag{2.3}$$

should be easier to solve than that corresponding to (2.2). Secondly, the form B should be spectrally equivalent to A in the sense that there are positive constants λ_0 and λ_1, with λ_1/λ_0 not too large, satisfying

$$\lambda_0 B(V, V) \leq A(V, V) \leq \lambda_1 B(V, V) \quad \text{for all } V \in \tilde{S}_h . \tag{2.4}$$

The problem of calculating the action of the inverse of the preconditioner that we shall describe will essentially reduce to the solution of discrete mixed problems on the refined subgrids, and discrete Neumann problems on the original grid. Due to the regularity of the mesh geometry, such problems are generally easier to solve than the system resulting from the composite grid discretization. Indeed, it is no longer necessary to store and utilize complicated connection structure and efficient vectorization results.

To motivate and describe the preconditioner we consider a certain decomposition of functions in S_h. Let $S_h^r(\Omega_2)$ be the collection of functions in S_h whose support is contained in Ψ_2. We decompose functions $V \in S_h$ by $V = V_P - V_H^r$, where $V_P \in S_h^r(\Omega_2)$ and V_H^r satisfies

$$A(V_H^r, \theta) = 0 \quad \text{for all } \theta \in S_h^r(\Omega_2) . \tag{2.5}$$

The function V_H^r equals V in Ω_1 and is 'discrete harmonic with respect to the refined grid' in Ω_2. The above decomposition is orthogonal with respect to the form A and hence

$$A(V, V) = A_1(V, V) + A_2(V_P, V_P) + A_2(V_H^r, V_H^r) , \tag{2.6}$$

where

$$A_i(u, v) \equiv \int_{\Omega_i} \frac{k}{\mu} \nabla u \cdot \nabla v \, dx .$$

Let $S_h^c(\Omega_2)$ be the collection of functions in S_h^c whose support is contained in Ω_2. We define the preconditioner by replacing V_H^r in the last term in (2.6) by the coarse-grid discrete harmonic function $V_H^c \in S_h^c$ defined by $V_H^c = V$ in Ω_1 and

$$A_2(V_H^c, \theta) = 0 \quad \text{for all } \theta \in S_h^c(\Omega_2) .$$

The form on $\tilde{S}_h$ corresponding to the preconditioner is thus defined by

$$B(V, V) = A_1(V, V) + A_2(V_P, V_P) + A_2(V_H^c, V_H^c) . \tag{2.7}$$

It is important to note here that although V_H^r does not explicitly appear on the right-hand side of (2.7), the term $A_2(V_H^c, V_H^c)$ implicitly defines a form acting on V_H^r. This is because V_H^c is completely determined from the values of V_H^r on Ω_1.

Using the properties of discrete harmonic functions it is not difficult to show that for this choice of B, (2.4) holds with $\lambda_1/\lambda_0 \leq C$, where C is a constant which is independent of the mesh size h (cf. [9]).

We next consider the problem of solving the system associated with the preconditioner. That is, given a function $g \in \tilde{S}_h$ we must compute the solution $W \in \tilde{S}_h$ satisfying

$$B(W, \theta) = (g, \theta) \quad \text{for all } \theta \in \tilde{S}_h . \tag{2.8}$$

The form B given by (2.7) on $\tilde{S}_h$ extends to S_h. Note that (2.8) holds for all functions in S_h. To compute the solution of (2.8), we shall first compute a function U in S_h satisfying (2.8) for all $\theta \in S_h$. Then the solution W of (2.8) is given by $U - \tilde{U}$, where $\tilde{U}$ is the mean value of U.

To calculate U, it suffices to compute U_p and U_H^r appearing in the decomposition $U = U_P + U_H^r$. Let $\theta \in S_h$ and similarly decompose $\theta = \theta_P + \theta_H^r$. Then U satisfies

$$A_1(U, \theta) + A_2(U_P, \theta_P) + A_2(U_H^c, \theta_H^c) = (g, \theta) \quad \text{for all } \theta \in S_h . \tag{2.9}$$

By considering only $\theta \in S_h^r(\Omega_2)$ ($\theta = 0$ on Ω_1 and hence $\theta_H^c = 0$) we get

$$A_2(U_P, \theta) = (g, \theta) \quad \text{for all } \theta \in S_h^r(\Omega_2) . \tag{2.10}$$

We use the discrete system (2.10) to solve for U_P; this corresponds to a mixed problem on the refinement subdomains. Since U is a solution of (2.9), U_H^c satisfies

$$A(U_H^c, \theta) = (g, \bar{\theta}) - A_2(U_P, \bar{\theta}) \quad \text{for all } \theta \in S_h^c , \tag{2.11}$$

where $\bar{\theta}$ is any function in S_h which equals θ on Ω_1. We use (2.11) to compute U_H^c; this problem corresponds to a discrete Neumann problem on the original coarse grid. We note that $U_H^r = U_H^c$ on Ω_1, hence we need only compute the values of U_H^r on Ω_2 (since $U = U_P + U_H^r$). To do this, let $\bar{U}$ be any function in S_h which equals U_H^c on Ω_1. Then $U_H^r = \bar{U} + \chi$, where χ is the unique function in $S_h^r(\Omega_2)$ satisfying

$$A_2(\chi, \theta) = -A_2(\bar{U}, \theta) \quad \text{for all } \theta \in S_H^r(\Omega_2) . \tag{2.12}$$

System (2.12) is a discrete mixed problem on the refinement subregions. The solution W of (2.8) is obtained from U by subtracting its mean value $\tilde{U}$.

We summarize the process by outlining the steps for obtaining a solution $U \in S_h$ of (2.9) and then the solution W of (2.3).

2.1. Algorithm for computing W

Step 1. Find U_P by solving mixed problems on the refinement subregions (2.10).
Step 2. Compute the data for (2.11) and then compute any solution U_H^c of the coarse-grid problem (2.11).
Step 3. Find U_H^r on Ω_2 by computing the discrete harmonic extension with respect to the refinement subspaces (this again involves solving mixed problems on the refinement subregions (2.12)).
Step 4. Compute $\tilde{U}$, the mean value of $U = U_P + U_H^r$. Set $W = U - \tilde{U}$.

REMARK 2.1. For clarity of presentation, we have avoided generality in the definition of the preconditioner. In particular, the preconditioner could be defined by replacing A in its construction with any other spectrally equivalent form $\mathscr{A}$. The form $\mathscr{A}$ could be chosen so that the resulting discrete systems could be efficiently solved (for example by 'fast solution techniques'). For a discussion of how this may be accomplished see [9].

We next consider some of the aspects of integrating the above preconditioning technique into existing programs. Except for trivial vector operations, the implementation of preconditioned conjugate gradient iteration only requires evaluation of the operator and inversion of the preconditioner. We shall demonstrate that codes for the above two tasks can be developed using standard evaluation and inversion subroutines for the coarse and refinement problems and some special procedures for handling the internal boundary variables (on the boundary between the coarse and refinement subregions).

Let us consider the data structure problem. As will become clear in further discussion, it is most convenient to represent functions in S_h in two storage modes. In the first mode, the values at the nodes (slave nodes not included) of the composite grid are stored in some order. The second mode stores nodal values in blocks associated with the coarse grid and refinement grids. The variables in these blocks appear in the natural ordering associated with the grid (noncomposite) structure. Thus the coarse-grid block includes variables for all points on the original coarse grid. The refined grid blocks include variables in the refined subsquares and their boundary. Data in mode-two vectors are redundant at nodes on the internal boundary and may be irrelevant on points of the coarse grid which are interior to the subsquares. The operator evaluation and preconditioner inversion section of the code most conveniently processes vectors in the second mode. The remaining segment of the preconditioned algorithm is readily available in subroutine form for vectors in the first mode. In implementation, it is not difficult or expensive to switch from one storage mode to another.

We now consider the problem of operator evaluation. Let $\{\varphi_i\}_{i=1}^N$ be the nodal basis for S_h. Given a function $g \in S_h$, we must compute

$$v_i = A(g, \varphi_i) \quad \text{for } i = 1, \ldots, N.$$

If g is a vector in the second mode, we compute v by first applying a coarse-grid evaluator on the coarse-grid segment of g to compute

$$A_1(g, \varphi_i).$$

We next apply a refined grid evaluator on the refined grid segments of g to compute

$$A_2(g, \varphi_i) .$$

Note that all of these evaluators operate on vectors of their respective grid types and consequently can take full advantage of any regular geometric structure of the subgrid.

We next consider the solution of (2.3). Let g be a vector in the second mode. Step 1 of the algorithm involves solving (2.10) for U_P which entails the refinement solution operators acting on the refinement blocks of g. Computing the data for Step 2 requires some new procedures which communicate between internal boundary portions of g. The remainder of Step 2 involves the solution of the full coarse problem (2.11) on a modified coarse block of g. Step 3 involves some additional communication between coarse and refinement boundary segments and the evaluation of the refinement solution operators on the (modified) refinement data segments. Thus we see that all of the respective solution processes act on the regular data structures for the coarse grid or the refinement grids.

3. Numerical results

In this section we provide numerical results which indicate how effective the previously described preconditioning techniques are on problems with singularites. We will first consider the question of how rapidly the preconditioned iteration converges in practice. Theoretically, we know that the condition numbers for the preconditioned systems remain independent of the number of unknowns. We shall see that the condition numbers for the examples of this section actually are more than two. This means that when applying a preconditioned conjugate gradient method, we are guaranteed an iterative error reduction of more than 0.17 per step in an appropriate norm [20]. We will also give examples which show how effective refinements of the form illustrated by Figs. 3 and 4 are in problems with singular behavior.

EXAMPLE 3.1. For the first example we will consider the variable coefficient where

$$\frac{k}{\mu}(x, y) = \{1 + 10(x^2 + y^2)\}^{-1} . \tag{3.1}$$

We will demonstrate the rate of iterative convergence for the preconditioned algorithm described in Section 2 in the two cases corresponding to refinements illustrated by Figs. 3 and 4. We shall indicate the iterative convergence rate in general by first computing the condition number K of the preconditioned system. We also give iterative convergence results for the method applied to the problem of approximating the solution of (1.1) in the case where the forcing function is given by $\delta(0, 0) - \delta(1, 1)$. For this problem we report the average reduction ρ in the discrete L^2 norm of the residual per iteration and the number n of iterations required to reduce this norm of the error by a factor of 0.0001.

REMARK 3.2. In this and the examples to follow, the solution has a logarithmic growth near the two corners $(0, 0)$ and $(1, 1)$. Accordingly, the mesh of Fig. 3 is chosen in a logarithmic

Table 1
Iterative convergence results for Example 3.1 with regular refinement

h	ρ	n	K
$\frac{1}{6}$	0.048	4	1.4
$\frac{1}{12}$	0.062	4	1.3
$\frac{1}{24}$	0.036	3	1.3

Table 2
Iterative convergence results for Example 3.1 with 'radial-like' refinement

h	ρ	n	K
$\frac{1}{6}$	0.063	4	1.3
$\frac{1}{12}$	0.082	4	1.4
$\frac{1}{24}$	0.092	4	1.5

fashion. More precisely, if r_i is the distance to the corner of the ith node along an edge, then $\ln(r_i) - \ln(r_{i+1}) = h_i$. The closest node to the corner is at a distance on the order of h^2.

Table 1 gives convergence results for the preconditioning algorithm applied to (3.1) using the regular refinements of Fig. 3, where the mesh size of the refinement is one half the size of h. The results for the 'radial-like' refinements of Fig. 4 are given in Table 2.

We remark that Table 1 gives results which are typical for experiments with the regular refinement preconditioner. We have tested examples where h(refined) was a quarter, an eight, and one sixteenth of h(coarse). In all examples, the condition number was less than two and the observed number of iterations were comparable.

EXAMPLE 3.3. This example will demonstrate the effectiveness of our refinement schemes on problems with logarithmic singularities. We shall consider the problem

$$-\Delta u = f \quad \text{on } \Omega \equiv [0,1]^2,$$

$$\frac{\partial u}{\partial n} = g \quad \text{on } \partial\Omega,$$

where Δ denotes the Laplacian ($\Delta \equiv \partial^2/\partial^2 x + \partial^2/\partial^2 y$) and n denotes the outward normal on $\partial\Omega$. For this problem, the function f was taken to be $\frac{1}{2}\pi[\delta(1,1) - \delta(0,0)]$ and g was chosen so that the resulting solution was $u = \frac{1}{2}(\ln|x| - \ln|(1,1) - x|)$. In these examples we shall be especially concerned with the effects that the refinement grids have on the approximation error.

Tables 3 and 4 give the discrete L^2 and L^∞ error on domain Ω_1 (ε_1 and ε_1^∞, respectively) as well as the discrete L^2 error on $\Omega_2/R_s(\varepsilon_2)$, where R_s is defined to be the union of the triangles of the refinement grids which intersect the singular points $(0,0)$ and $(1,1)$. The discrete L^2 errors on Ω_2 corresponding to the two different types of refinements may be somewhat difficult to compare since the radial-like grids measure errors much closer to the singularities. We also include n, the number of iterations required to reduce a discrete L^2 norm of the residual iteration error by 0.0001. We finally include N, the number of unknowns in the largest system solved in the inversion of the preconditioner (see Section 2). As previously observed, these systems correspond to the refinement and coarse grids and always have a regular topological structure. Table 5 indicates corresponding convergence results with no local refinement.

In our opinion, the radial-like refinement technique is the most advantageous from the accuracy as well as the programming point of view. We see from the tables that the radial-like refinement scheme converges somewhat faster than the regular refinement scheme. A much

Table 3
Convergence results for Example 3.3 with 'radial-like' refinement

h	n	ε_1	ε_2	ε_1^∞	N
$\frac{1}{6}$	3	4.3×10^{-3}	1.2×10^{-2}	1.2×10^{-2}	35
$\frac{1}{12}$	4	1.3×10^{-3}	2.8×10^{-3}	3.6×10^{-3}	171
$\frac{1}{24}$	4	3.0×10^{-4}	7.8×10^{-4}	1.1×10^{-3}	833

Table 4
Convergence results for Example 3.3 with a regular refinement and h(refinement) = h(coarse)/4

h	n	ε_1	ε_2	ε_1^∞	N
$\frac{1}{6}$	4	4.4×10^{-3}	1.2×10^{-2}	1.2×10^{-2}	81
$\frac{1}{12}$	5	1.9×10^{-3}	4.3×10^{-3}	6.0×10^{-3}	289
$\frac{1}{24}$	4	1.0×10^{-3}	1.9×10^{-3}	3.0×10^{-3}	1089

Table 5
Convergence results for Example 3.3 with no refinement

h	n	ε_1	ε_2	ε_1^∞	N
$\frac{1}{6}$	1	1.2×10^{-2}	1.4×10^{-2}	2.5×10^{-2}	49
$\frac{1}{12}$	1	6.9×10^{-3}	9.7×10^{-3}	1.6×10^{-2}	169
$\frac{1}{24}$	1	3.6×10^{-3}	5.4×10^{-3}	9.0×10^{-3}	625
$\frac{1}{48}$	1	1.9×10^{-3}	3.0×10^{-3}	5.0×10^{-3}	2401

more drastic improvement would be noticed if smaller h's were run since the radial-like refinement shows second-order convergence as opposed to the first-order convergence of the regular refinement scheme. From the programming point of view, in coding the radial refinement scheme, one must keep track of the refinement grid vertices. This is only a minor complication. In contrast, the introduction of slave nodes in the regular refinement scheme leads to much more significant programming and logic problems. The connection-communication routines between nodes near the coarse refinement boundary are much more complicated to code and debug for the regular refinement scheme.

References

[1] I. Babuška and W.C. Rheinboldt, A posteriori error estimates for the finite element method, Internat. J. Numer. Meths. Engrg. 12 (1978) 1597–1615.
[2] I. Babuška and W.C. Rheinboldt, Reliable error estimation and mesh adaptation, in: J.T. Oden, ed., Computational Methods in Nonlinear Mechanics (North-Holland, New York, 1984).
[3] R.E. Bank and A.H. Sherman, PLTMG user's guide, Tech. Rept. No. 152, Center for Numerical Analysis, University of Texas at Austin, Austin, TX, 1979.
[4] R.E. Bank and A. Weiser, Some a-posteriori error estimates for elliptic partial differential equations, Math. Comp. 44 (1985) 283–301.
[5] M.J. Berger, Data structures for adaptive mesh refinement, in: I. Babuška, J. Chandra and J.E. Flaherty,

eds., Adaptive Computational Methods for Partial Differential Equations (SIAM, Philadelphia, PA, 1983) 237–251.

[6] P.E. Bjørstad and O.B. Widlund, Iterative methods for the solution of elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal. 23 (1986) 1097–1120.

[7] P.E. Bjørstad and O.B. Widlund, Solving elliptic problems on regions partitioned into substructures, in: G. Birkhoff and A. Schoenstadt, eds., Elliptic Problem Solvers II (Academic Press, New York, 1984) 245–256.

[8] J.H. Bramble, J.E. Pasciak and A.H. Schatz, An iterative method for elliptic problems on regions partitioned into substructures, Math. Comp. 46 (1986) 361–369.

[9] J.H. Bramble, J.E. Pasciak and A.H. Schatz, The construction of preconditioner for elliptic problems by substructing, I., Math. Comp 47 (1986) 103–134.

[10] R.J. Charbenau and R.L. Street, Modeling groundwater flow fields containing point singularities: A technique for singularity removal, Water Resources Res. 15 (1979) 583–599.

[11] B.L. Darlow, R.E. Ewing and M.F. Wheeler, Mixed finite elements methods for miscible displacement in porous media, Soc. Pet. Engrg. J. 4 (1984) 391–398.

[12] L. Demkowicz, J.T. Oden and P. Devloo, An *h*-type mesh refinement strategy based on a minimization of interpolation error, Comput. Meths. Appl. Mech. Engrg. 53 (1985) 67–89.

[13] J.C. Diaz, R.E. Ewing, R.W. Jones, A.E. McDonald, L.M. Whler and D.U. von Rosenberg, Self-adaptive local grid refinement for time dependent, two dimensional simulation, in: R.H. Gallagher, G. Carey, J.T. Oden and O.C. Zienkiewicz, eds., Finite Elements in Fluids 4 (Wiley, New York, 1985) 279–290.

[14] R.E. Ewing, J.V. Koebbe, R. Gonzalez and M.F. Wheeler, Mixed finite element methods for accurate fluid velocities in: Finite Elements in Fluids 4 (Wiley, New York, 1985) 233–249.

[15] R.E. Ewing, Adaptive mesh refinement in large scale fluid flow simulation, in: I. Babuška, O.C. Zienkiewicz and E. Arantes de Oliveira, eds., Accuracy Estimates and Adaptivity for Finite Elements (Wiley, New York, 1986) 299–314.

[16] R.E. Ewing, Efficient adaptive procedures for fluid flow applications, Comput. Meths. Appl. Mech. Engrg. (to appear).

[17] J.E. Flaherty, J.M. Coyle, R. Ludwig and S.F. Davis, Adaptive finite element methods for parabolic partial differential equations, in: I. Babuška, J. Chandra and J.E. Flaherty, eds., Adaptive Computational Methods for Partial Differential Equations (SIAM, Philadelphia, PA, 1983) 144.

[18] G.H. Golub and D. Meyers, The use of preconditioning over irregular regions, in: Proceedings of the Sixth International Conference on Computing Methods in Science and Engineering, Versailles, France, 1983.

[19] J.T. Oden, T. Strouboulis and P. Devloo, Adaptive finite element methods for the analysis of inviscid compressible flow, 1. Fast refinement/unrefinement and moving mesh methods for unstructured meshes, Comput. Meths. Appl. Mech. Engrg. (to appear).

[20] W.M. Patterson, 3rd, Iterative Methods for the Solution of a Linear Operator Equation in Hilbert Space—a Survey, Lecture Notes in Mathematics 394 (Springer, New York, 1984).

[21] W.C. Rheinboldt and C.K. Mesztenyi, On a data structure for adaptive finite element mesh refinement, Trans. Math. Software 6 (1980) 166–187.

[22] A.H. Schatz and L.B. Wahlbin, Maximum norm estimates in the finite element method on plain polygonal domains, Part II, refinements, Math. Comp. 33 (1979) 465–492.

[23] D.U. von Rosenberg, Local mesh refinement for finite difference methods, paper SPE 10974, presented at SPE Annual Technical Conference and Exhibition, New Orleans, LA, 1982.