ADAPTIVE GRIDDING TECHNIQUES FOR GROUNDWATER CONTAMINANT MODELING (COLLECTION OF PUBLICATIONS)

Myron B. Allen

1990
Final Report WWRC-90-33

Final Report

Submitted to

Wyoming Water Research Center
University of Wyoming
Laramie, Wyoming

Submitted by

Myron B. Allen
Department of Mathematics
University of Wyoming
Laramie, Wyoming

Table of Contents

1. TECHNICAL SYNOPSIS

This section of the report is a brief synopsis of the project's scientific aims and accomplishments. The discussion in this section is intended for a technical audience, but it does not assume that readers are specialists in mathematical modeling. The Appendix to this report, summarized in Section 2, consists of published scientific articles that describe the results of the project for specialists.

1.1 Objective

The original objective of the project was to develop numerical techniques for modeling groundwater contaminant flows in the presence of sharp fronts in contaminant concentration. Such fronts occur and persist in contaminant flows in which the spreading attributable to hydrodynamic dispersion is small compared with advective transport along the groundwater velocity field. This "advection-dominated" transport regime is well documented in the water resources literature.

Steep concentration fronts in advection-dominated flows pose severe problems for most standard numerical models. Such models usually rely on approximation schemes in which one treats the real continuous aquifer as a discrete network of cells or nodes, called a grid. In each cell, the model assumes that concentrations, velocities, and other cell variables vary in a simple fashion. For example, these quantities may be constant over each cell. When a steep front is present, many small cells are needed in the vicinity of the front to produce accurate approximations of the local variations in contaminant concentration. Since the cost of running a model increases with the number of cells used, it is useful to be able to use small cells–that is, to refine the grid—only in the small regions near the fronts, where improved resolution is needed.

Installing this capability in acutal computer codes is a challenging task. Since contaminant fronts move, the regions of refined grid must move adaptively as well. Mathematically, moving a zone of locally refined grid changes the algebraic relationships among the cell variables in a complicated manner that one cannot predict in advance of running the model. In contrast with the case when a single, coarse grid is adequate, grids having moving zones of local refinement require innovative algorithmic structures if they are to be computationally efficient. The purpose of this work has been to develop such structures.

1.2 Related applications

Adpative local grid refinement has applications in a wide array of fluid-dynamic settings. In the field of groundwater contamination, adaptive local grid refinement is useful in a variety of problems beside the problem of passive solute transport. Of special interest are multiphase flows, such as air-water flows in the vadose zone or flows involving nonaqueous-phase liquids (NAPLs), where steep fronts or even shocks in phase saturations commonly arise.

1.3 Summary of accomplishments

Early in the project, considerable effort focused on adaptive gridding techniques for contaminant transport in one space dimension. We devised a finite-element collocation scheme

that is quite effective in that setting and that is readily amenable to implementation on parallel-processing computers (Allen and Curran, 1989). However, that scheme does not readily extend to problems in higher dimensions.

We also investigated a class of methods for two-dimensional problems using highly parallelizable, alternating-direction collocation schemes (Curran and Allen, 1989 and 1990; Allen and Khosravani, 1990; Khosravani, 1989; Li, 1990). As part of this effort, we collaborated with researchers at the University of Vermont, sponsoring a week-long visit to Wyoming that culminated in the development of a parallelizable alternating-direction scheme suitable for tensor hydrodynamic dispersion (Guarnaccia and Pinder, 1989).

To implement grid refinement in these two-dimensional codes, we revisited the one-dimensional case, devising a scheme that extends readily to the alternating-direction setting (Curran Allen, in preparation). The actual implementation of this technique is the subject of a Ph.D. dissertation in August, 1990 (Curran, 1990).

1.4 Ongoing work

The development of accurate and efficient contaminant transport codes leads naturally to the study of the effects of aquifer heterogeneity, a topic of much current interest in the water resources community. Accurate transport models enable one to study the numerical problem of scaling up from small-scale heterogeneities in an aquifer to the scales comparable to practical cell diameters in numerical models. During the course of this project, we began to outline some of these considerations (Allen and Ewing, 1990) and initiated research into numerical schemes for groundwater flow that will complement our transport codes (Allen et al., submitted).

2. PUBLICATIONS RESULTING FROM THE WORK

The following is a list of publications that grew out of the work. This list serves as a bibliography for Section 1. Copies appear in the Appendix, except for M.S. and Ph.D. work, which are available through the University of Wyoming.

2.1 Refereed articles

- Allen, M. B., and Curran, M. C. (1989), "Adaptive local grid refinement algorithms for finite-element collocation," *Numer. Math. P.D.E.*, **5**, 121–132.
- Curran, M.C., and Allen, M.B. (1990), "Parallel computing for solute transport models via alternating-direction collocation," *Adv. Water Resovr.*, **13:2**, 70–75.
- Allen, M. B., Ewing, R. C., and Lu, P. (submitted), "Well conditioned iterative schemes for mixed finite-element models of porous-media flows."
- Curran, M. C., and Allen, M. B. (in preparation), "A domain-decomposition approach to local grid refinement in finite-element collocation."

2.2 Papers in conference proceedings

- Curran, M. C., and Allen, M. B. (1989; invited) "Parallel computing speedups for alternating direction collocation," in *Finite Element Analysis Fluids: Proceedings of the Seventh International Conference*, Huntsville, Alabama, ed. by T. J. Chung and G. R. Karr, Huntsville, AL: UAH Press, 947–952.
- Guarnaccia, J. F., and Pinder, G. F. (1989), "A parallel collocation based algorithm for the generalized transport equation," in *Applications of Supercomputers in Engineering, Proceedings of the First International Conference*, Southhampton, U.K., ed. by C. A. Brebbia and A. Peters, Amsterdam: Elsevier.
- Allen, M. B., and Ewing, R. E. (1990), "How aquifer heterogeneities affect numerical groundwater models," in *Proceedings, Groundwater Engineering and Management Conference*, organized by Colorado Water Resources Research Institute and Office of the State Engineer, Denver, CO, February 28–March 1, 1990, 161-170.
- Allen, M. B., and Khosravani, A. (1990), "An Eulerian-Lagrangian method for finite-element collocation using the modified method of characteristics," in *Proceedings, Eighth Interantional Conference in Water Resources*, Venice, Italy, ed. by G. Gambolati et al., Southhampton, U. K.: Computational Mechanics Publications, 375–379.

2.3 M.S. and Ph.D. work

- Khosravani, A. (1989), "Numerical Solutions of Solute Transport Equations," M.S. paper, Department of Mathematics, University of Wyoming, Laramie, WY, October, 1989.
- Li, X (1990), "Numerical Methods for the Advection-Diffusion Equation in Areally and Vertically Averaged Domains," M.S. paper, Department of Mathematics, University of Wyoming, Laramie, WY, July, 1990.
- Curran, M. C. (1990), "Numerical Schemes for Highly Advective Flows Using Finite-Element Collocation with Adaptive Local Grid Refinement", Ph.D. dissertation, Department of Mathematics, University of Wyoming, Laramie, WY, August, 1990.

2.4 User's guide

- Khosravani, A., and Allen, M. B., "User's Guide to ADMOC," University of Wyoming Department of Mathematics, Laramie, WY, August, 1990.

3. GRADUATE STUDENT TRAINING

Four graduate students in Mathematics received partial support from this project. Three of these students completed degrees during the course of the project:

- Azar Khosravani, M.S., 1989
- Xingjing Li, M.S., 1990
- Mark C. Curran, Ph.D., 1990

The fourth student, Yun Li, began working on research for the M.S. in Mathematics in June, 1990, and anticipates completion of the degree during the 1990–91 academic year.

APPENDIX: COPIES OF PUBLICATIONS

Attached are copies of papers appearing in or submitted to refereed journals or presented at conferences. Also attached is a user's guide to a transport code. Not attached are Mark Curran's Ph.D. dissertation, which is available from the University of Wyoming Library, and the M.S. papers written by Azar Khosravani and Xingjing Li, which are on file at the Department of Mathematics, University of Wyoming.

Adaptive Local Grid Refinement Algorithms for Finite-Element Collocation

Myron B. Allen and Mark C. Curran
Department of Mathematics, University of Wyoming, Laramie, Wyoming 82071

An adaptive grid refinement procedure allows accurate solutions to advection-dominated, time-dependent flows using finite-element collocation. The technique relies on a data structure that is readily amenable to parallel computing. The paper discusses computational aspects of the method.

I. INTRODUCTION

Adaptive gridding offers an important class of strategies for computing accurate solutions to highly advective fluid flows. We present an adaptive local grid refinement scheme for use in finite-element collocation models for such flows. Of special interest here are the algorithmic aspects of the procedure, which is readily amenable to implementation on parallel-architecture computers. We focus on transient flows in one space dimension. The paper has the following structure: Section II briefly reviews finite-element collocation on fixed grids; Section III discusses the grid-refinement algorithm for the linear advection-diffusion equation; Section IV extends the algorithm to nonlinear problems using Burgers' equation as an example; Section V concludes the paper with an examination of the method's performance on a parallel computer.

II. REVIEW OF FINITE-ELEMENT COLLOCATION

The method of finite-element collocation has its roots in the engineering literature of the 1930s (see [1]), but we owe the modern version to de Boor and Swartz [2] and Douglas and Dupont [3], among others. For purposes of illustration, consider the constant-coefficient advection-diffusion problem posed on the spatial domain $\Omega = (0, L)$:

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} - D\frac{\partial^2 u}{\partial x^2} = 0, \qquad (x,t) \in \Omega \times (0,\infty), \tag{1a}$$

$$u(x,0) = u_I(x), \qquad x \in \Omega, \tag{1b}$$

$$u(0,t) = u_0, \qquad \frac{\partial u}{\partial x}(L,t) = u'_N, \qquad t \geq 0. \tag{1c}$$

Numerical Methods for Partial Differential Equations, 5, 121–132 (1989)

CCC 0749-159X/89/05121-12$04.00

Here, $v > 0$ represents fluid velocity, $D > 0$ is a diffusion coefficient, and $u = u(x, t)$ stands for an unknown function, say, solute concentration. We shall apply finite-element collocation to the Crank-Nicolson semidiscrete analog

$$u^{n+1} - u^n + k\left(v\frac{du^{n+(1/2)}}{dx} - D\frac{d^2u^{n+(1/2)}}{dx^2}\right) = 0,$$

where the superscripts indicate time level, $(\cdot)^{n+(1/2)} \equiv \frac{1}{2}[(\cdot)^{n+1} + (\cdot)^n]$, and k signifies the time step.

We begin by establishing a spatial grid $\Delta^0 = \{0 = x_0, h = x_1, \cdots, Nh = x_N = L\}$, and call $[x_{i-1}, x_i] = \overline{\Omega}_i$. In later sections, Δ^0 will be the *coarse grid*, and $\overline{\Omega}_i$ will be the *i*th *coarse-grid element*. The space of Hermite piecewise cubics for the grid Δ^0 on $\overline{\Omega} = [0, 1]$ is

$$\mathcal{M}_1^3(\Delta^0) = \{f \in C^1(\overline{\Omega}) | f|\overline{\Omega}_i \text{ is cubic}\}.$$

In other words, f is cubic on each subinterval $\overline{\Omega}_i$ and, globally, is continuously differentiable. This order of continuity is the lowest for which one can use collocation on a second-order differential equation (Birkhoff and Lynch [4], p. 200).

The space $\mathcal{M}_1^3(\Delta^0)$ has an interpolating basis $\{H_{i,0}, H_{i,1}\}_{i=0}^N$ in which the support of each function $H_{i,j}(x)$ is a small subset of $\overline{\Omega} = [0, L]$ consisting of at most two adjacent subintervals, $\overline{\Omega}_{i-1} \cup \overline{\Omega}_i$ (Prenter [5], Chapter 3). In terms of this basis, we can write any $f_0 \in \mathcal{M}_1^3(\Delta^0)$ as a linear combination involving values of f and f' at the nodes of Δ^0:

$$f(x) = \sum_{i=0}^{N} [f(x_i)H_{0,i}(x) + f'(x_i)H_{1,i}(x)].$$

In fact, for any $g \in C^1(\overline{\Omega})$, we can define a projection onto $\mathcal{M}_1^3(\Delta^0)$ as

$$(\pi^0 g)(x) = \sum_{i=0}^{N} [g(x_i)H_{0,i}(x) + g'(x_i)H_{1,i}(x)].$$

To solve the semidiscrete analog of the problem (1), we determine a sequence $\{\hat{u}^n\}_{n=0}^{\infty}$ by first imposing initial and boundary conditions:

$$\hat{u}^0(x) = \pi^0 u_I(x) \quad \forall\, x \in \overline{\Omega};$$

$$\hat{u}^n(0) = u_0; \qquad \frac{d\hat{u}^n}{dx}(L) = u_N', \qquad n = 1, 2, \cdots.$$

These criteria specify $\hat{u}^0$ completely and determine two of the $2N \times 2$ nodal degrees of freedom for $\hat{u}^1, \hat{u}^2, \cdots$. To determine the remaining $2N$ degrees of freedom at each time level $n + 1$, we first form the residual

$$R^{n+1} = \hat{u}^{n+1} - \hat{u}^n + k\left(v\frac{d\hat{u}^{n+(1/2)}}{dx} - D\frac{d^2\hat{u}^{n+(1/2)}}{dx^2}\right).$$

We then pick a collection $\{\bar{x}_1, \cdots, \bar{x}_{2N}\} \subset \Omega$ *of collocation points* and force $R^{n+1}(\bar{x}_k) = 0, k = 1, \cdots, 2N$. Douglas and Dupont [3] show that one can obtain optimal-order error estimates of the form $\|\hat{u}^n - u(\cdot, nk)\|_\infty = \mathcal{O}(h^4)$ by choosing the $\bar{x}_k$ to be the two-point Gauss-quadrature abscissae in each element Ω_i.

Allen and Pinder [6] demonstrate an upstream-weighted technique assigning precisely these collocation points to all terms in the residual except the advection terms $kvd\hat{u}^n/dx$, for which the "collocation" points have the form $\bar{x}_k^* = \bar{x}_k - \zeta h, \zeta > 0$.

Despite the smoothness required of the trial function $\hat{u}^n$, two features of collocation make it an attractive scheme for modeling transient, advection-dominated flows. First, the matrix for the system of collocation equations at each time level has bandwidth five in one space dimension and is therefore sparser than the matrices arising from other fourth-order finite-element schemes. The price paid for this sparseness is a loss of symmetry in the matrix equations approximating self-adjoint problems—a penalty that is irrelevant in advective problems, since they are generally nonself-adjoint. Second, in contrast with classical Galerkin formulations, computing the collocation matrix requires neither the calculation of integrals nor formal assembly of a global matrix from local element matrices. This latter fact makes the method especially useful in transient, nonlinear problems, which typically require the computation of a new matrix at each iteration of each time step.

III. THE ADVECTION-DIFFUSION EQUATION

Finite element collocation, like other discrete methods, tends to yield unacceptable results for the advection-diffusion equation when the Peclet number $P = vL/D \gg 1$. In its standard $\mathcal{O}(h^4)$ version, collocation yields spuriously oscillatory solutions near sharp fronts unless $h < \sqrt{12}/P$ (Jensen and Finlayson [7]). On the other hand, the upstream collocation scheme just cited smears sharp fronts as a consequence of a numerical diffusion coefficient proportional to $Ph\zeta$ (Allen [8]). Figure 1 illustrates these types of error. When $P \gg 1$, using a uniform grid Δ^0 fine enough to mitigate these errors can be expensive. One way around this dilemma is to adjust h *locally*, so that the grid spacing is small only in regions where the solution exhibits sharp fronts needing fine-scale spatial resoluton. Since the sharp fronts move, it is necessary to refine the grid *adaptively*, so that the refined zone follows the front.

Toward this end, we construct a sequence $\{\Delta^n\}_{n=0}^{\infty}$ of grids, each associated with a time level n. For computational convenience we demand that each $\Delta^n \supset \Delta^0$, so that the variables associated with the original coarse grid Δ^0 are present at every time level. Thus at each time level n we construct a mapping ν^n: $\{1, \cdots, N\} \rightarrow \{0, 1, 2, \cdots\}$ assigning $\nu^n(i)$ new nodes, assumed evenly spaced, to each coarse-grid element $\bar{\Omega}_i = [x_{i-1}, x_i]$ formed by Δ^0. To avoid unnecessary computational effort, we want $\nu^n(i) = 0$ except when $\bar{\Omega}_i$ lies near a sharp front. In these exceptional cases, we determine $\nu^n(i)$ according to a grid-refinement strategy appropriate for the equation being solved. We denote by $Z^n = \sum_{i=1}^{N} \nu^n(i)$ the total number of new nodes added at time level n. Also, we associate with each grid Δ^n a trial space $\mathcal{M}_1^3(\Delta^n)$ and a corresponding projection π^n: $C^1(\bar{\Omega}) \rightarrow \mathcal{M}_1^3(\Delta^n)$ mapping continuously differentiable functions onto that trial space. Since the polynomial degree of the finite-element approximation remains constant while the grid spacing changes, this scheme is an example of *h-refinement*.

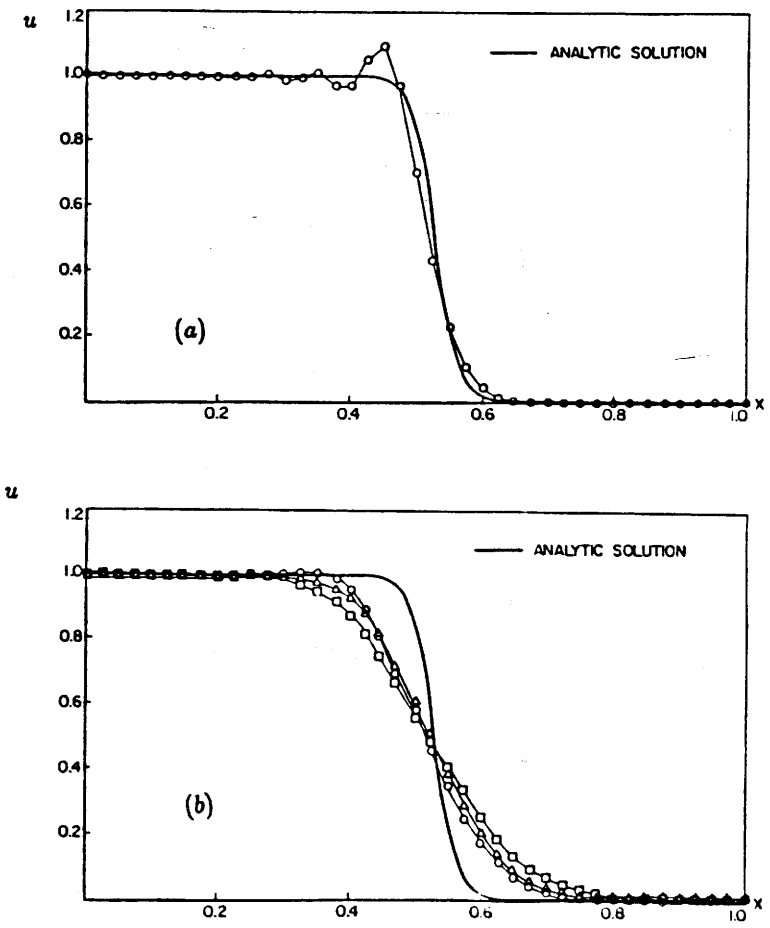


FIG. 1. Spurious oscillations and numerical diffusion associated with (a) standard and (b) upstream-weighted collocation solutions to the advection-diffusion equation. In all cases, $h = 1/40$, $k = 1/20$, $P = 1069$.

We now collocate as before to determine a sequence

$$\{\hat{u} \in \mathcal{M}_1^3(\Delta^0), \hat{u}^1 \in \mathcal{M}_1^3(\Delta^1), \cdots\},$$

using the $2(N + Z^{n+1})$ Gauss abscissae for Δ^{n+1} as collocation points to solve for the unknown Hermite coordinates of $\hat{u}^{n+1}$. One new wrinkle is that we must project the old solution $\hat{u}^n \in \mathcal{M}_1^3(\Delta^n)$ forward to the new trial space $\mathcal{M}_1^3(\Delta^{n+1})$ to form the residual, getting collocation equations that have the form

$$\hat{u}^{n+1}(\bar{x}_k) + \frac{k}{2}\left[\nu \frac{d\hat{u}^{n+1}}{dx}(\bar{x}_k) - D\frac{d^2\hat{u}^{n+1}}{dx^2}(\bar{x}_k)\right]$$
$$= (\pi^{n+1}\hat{u}^n)(\bar{x}_k) - \frac{k}{2}\left[\nu \frac{d}{dx}(\pi^{n+1}\hat{u}^n)(\bar{x}_k) - D\frac{d^2}{dx^2}(\pi^{n+1}\hat{u}^n)(\bar{x}_k)\right].$$

There is another new wrinkle. The addition of Z^n new nodes, and hence $2Z^n$ new unknowns and equations, disrupts the matrix structure associated with collocation on Δ^0. If we have an efficient matrix solver for the structure associated with Δ^0, then it makes sense to decouple the equations associated with newly added nodal parameters of $\hat{u}^{n+1}$, leaving a system having the original structure for the $2N$ coarse-grid unknowns along with a set of smaller systems for the $2Z^{n+1}$ new unknowns the construction of a *p-refinement* scheme for collocation, in which they improve spatial resolution by increasing the local polynomial degree of the aproximation.

We accomplish the decoupling in an elementwise fashion, using sparse row reduction on each of the augmented equation sets associated with refined coarse-grid elements $\overline{\Omega}_i$. At a typical time level $n+1$, the procedure, which we call *elementwise condensation*, yields a system of the form

$$\begin{bmatrix} \mathbf{A}_0^{n+1} & \mathbf{0} \\ \mathbf{B}^{n+1} & \mathbf{A}_R^{n+1} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0^{n+1} \\ \mathbf{u}_r^{n+1} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{r}}_0^{n+1} \\ \tilde{\mathbf{r}}_r^{n+1} \end{bmatrix}, \tag{2}$$

where $\mathbf{u}_0^{n+1} \in \mathbb{R}^{2N}$ denotes the vector of coarse-grid unknowns; $\mathbf{u}_r^{n+1} \in \mathbb{R}^{2Z^{n+1}}$ denotes the vector of refinement unknowns; $\mathbf{A}_R^{n+1} \in \mathbb{R}^{2Z^{n+1} \times 2Z^{n+1}}$ is an upper bidiagonal matrix multiplying the refinement unknowns, and $\mathbf{B}^{n+1} \in \mathbb{R}^{2Z^{n+1} \times 2N}$ is the matrix coupling new unknowns to the coarse-grid values. In practice, $\mathbf{A}_0^{n+1}$ has the same size and zero structure as the matrix associated with collocation on Δ^0, and $\mathbf{B}^{n+1}$ is sparse, having one $2\nu^{n+1}(i) \times 4$ nonzero block for every refined element $\overline{\Omega}_i$. Figure 2 shows the block structure of Eq. (2) in more detail.

Given this structure, we can solve for the vector $\mathbf{u}_0^{n+1}$ of coarse-grid variables using our efficient coarse-grid solver, then solve for the refinement unknowns essentially using back substitution via the coupling block $\mathbf{B}^{n+1}$. The time-stepping procedure, starting with $\hat{u}^n$ known, is as follows:

1. Compute $\nu^{n+1}(i)$, $i = 1, \cdots, N$, using an adaptive refinement strategy.
2. Form the projection $\pi^{n+1}\hat{u}^n$.

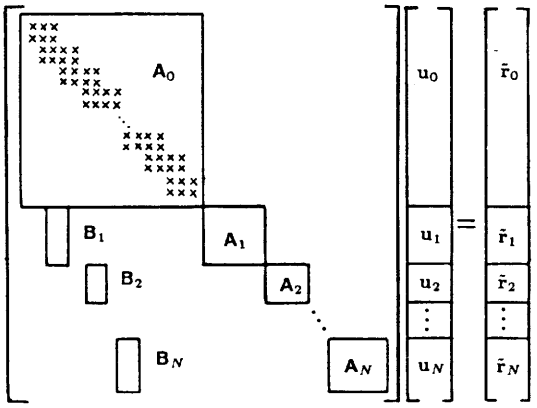


FIG. 2. Block structure of the matrix equation for the locally refined system after elementwise condensation.

3. Compute the matrix entries associated with the refined problem.
4. Use elementwise condensation to construct the system (2).
5. Solve $\mathbf{A}_0^{n+1}\mathbf{u}_0^{n+1} = \tilde{\mathbf{r}}_0^{n+1}$ for coarse-grid values.
6. Solve $\mathbf{B}\mathbf{u}_0^{n+1} + \mathbf{A}_R^{n+1}\mathbf{u}_r^{n+1} = \tilde{\mathbf{r}}_0^{n+1}$ for variables introduced by the refinement.

Step 6 actually reduces to a set of decoupled problems, each of which has the form

$$\mathbf{B}_i^{n+1}\begin{bmatrix} u_{i-1} \\ u'_{i-1} \\ u_i \\ u'_i \end{bmatrix}^{n+1} + \mathbf{A}_i^{n+1}\mathbf{u}_i^{n+1} = \tilde{\mathbf{r}}_i^{n+1}, \tag{3}$$

for a particular refined coarse-grid element $\overline{\Omega}_i$. Here, $\mathbf{B}_i^{n+1} \in \mathbb{R}^{2\nu^{n+1}(i)\times 4}$ multiplies the coarse-grid unknowns in $\overline{\Omega}_i$, and the upper bidiagonal matrix $\mathbf{A}_i^{n+1} \in \mathbb{R}^{2\nu^{n+1}(i)\times 2\nu^{n+1}(i)}$ multiplies the refinement unknowns in $\overline{\Omega}_i$. Observe that the back substitutions (3) associated with different refined elements $\overline{\Omega}_i$ are independent and therefore are amenable to concurrent processing. Similarly, the elementwise tasks called for in steps 1, 2, 3, and 4 are also parallelizable. We explore this aspect of the method in Section V.

A sample computation demonstrates the effectiveness of this procedure in yielding accurate simulations. Consider the problem (1) on $\Omega = (0, 1)$ with square-wave data,

$$u_I(x) = 0, \qquad u_0 = 1, \qquad u'_N = 0,$$

when $\nu = 0.369$ and $D = 0.001$. If we use a coarse gird Δ^0 having $h = k = 0.05$ and employ upstream weighting with $\zeta = 0.2$, then the numerical solution will exhibit significant smearing, as shown in Figure 3 for $t = 1$. We can virtually eliminate this smearing by forcing $h < 1/P$ globally, but as Figure 3 also shows, we can achieve comparable results by enforcing the same criterion only in zones where $\sup_{x\in\overline{\Omega}_i}|d\hat{u}^n/dx| > (5h)^{-1}$, that is, where the solution is steep. The latter strategy involves solving for at most 180 unknowns per time step, while global refinement requires solving for about 400.

IV. BURGERS' EQUATION

For nonlinear problems the time-stepping procedure is somewhat more complicated. Here, the use of an implicit scheme for stability forces one to iterate between time steps. Since frontal velocities may be functions of the unknown solution, it is possible that zones needing refinement at a particular time level will be identifiable only in the last few iterations of the time step, when the iterative scheme has nearly converged. We use this reasoning in developing a grid-refinement algorithm for Burgers' equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{\mu}{2}\frac{\partial^2 u}{\partial x^2} = 0, \qquad (x, t) \in \Omega \times (0, \infty),$$

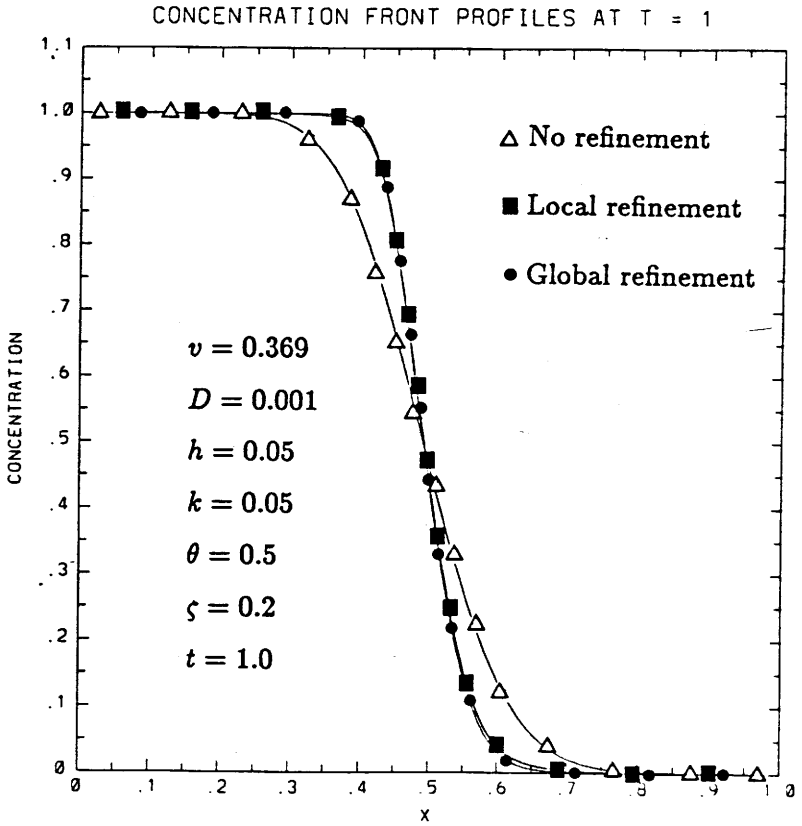


FIG. 3. Upstream-weighted collocation solutions to the advection-diffusion equation using a coarse grid, a globally refined grid, and a locally refined grid.

assuming initial and boundary data having the form

$$u(x,0) = u_I(x), \qquad x \in \Omega = (0,1),$$
$$u(0,t) = u_0, \qquad u(L,t) = u_N.$$

In this equation, u stands for fluid velocity, while μ represents a fluid viscosity. When $\mu \ll 1$, the equation models nearly inviscid, self-advected flows and has shock-like solutions needing local fine-scale spatial resolution.

In the refined problem on Δ^0, we compute a sequence $\{\hat{u}^n\}_{n=0}^{\infty}$ in $\mathcal{M}_1^3(\Delta^0)$, satisfying the initial and boundary conditions, such that the residual vanishes at each collocation point $\bar{x}_k \in \Omega$. In this case, the residual for the semidiscrete scheme is

$$R^{n+1} = \hat{u}^{n+1} - \hat{u}^n + k\left(\hat{u}^{n+1}\frac{d\hat{u}^{n+1}}{dx} - \frac{\mu}{2}\frac{d^2\hat{u}^{n+1}}{dx^2}\right),$$

which is a nonlinear function of the unknown Hermite coordinates

$$\{(u_0')^{n+1}, (u_1)^{n+1}, (u_1')^{n+1}, \cdots, (u_N')^{n+1}\}$$

for each fixed value of $\bar{x}_k$. To solve this nonlinear problem for $\hat{u}^{n+1}$ in terms of $\hat{u}^n$, we linearize it using Newton's method. Thus we make an initial guess $\hat{u}^{n+1,0} = \hat{u}^n$ and, at each iterative level $m > 0$, solve for a new iterate

$$\hat{u}^{n+1,m+1} = \sum_{i=0}^{N} \{[(u_i)^{n+1,m} + \delta_i]H_{0,i} + [(u_i')^{n+1,m} + \delta_i']H_{1,i}\}.$$

Clearly $\delta_0 = \delta_N = 0$; the boundary values of $\hat{u}^{n+1}$ are known. To compute the vector $\boldsymbol{\delta}$ of remaining increments, we solve the linear system

$$\mathbf{J}^{n+1,m}\boldsymbol{\delta} = -\mathbf{r}^{n+1,m},$$

where the kth entry of $\mathbf{r}^{n+1,m}$ is $R^{n+1,m}(\bar{x}_k)$, and $\mathbf{J}^{n+1,m}$ is the Jacobian matrix of $\mathbf{r}^{n+1,m}$ with respect to the unknown Hermite coordinates. Given a tolerance $\tau > 0$, we iterate until $\|\mathbf{r}^{n+1,m+1}\|_\infty < \tau$, then set $\mathbf{u}^{n+1,m+1}$.

In practice this scheme has several nice attributes. First, it is stable for very large time steps, including "Courant" numbers $\|\hat{u}^{n+1}\|_\infty k/h > 100$ that far exceed those required to keep the temporal truncation error reasonably small. Second, it converges rapidly. Using $N = 100$, the scheme reaches $\|\mathbf{r}^{n+1,m+1}\|_\infty < 10^{-7}$ in three or four interations, almost independent of the time step k.

To implement adaptive local grid refinement, we adopt a simple "predictor-corrector" strategy in this Newton scheme. This strategy determines the refined grid Δ^{n+1} only after performing a few Newton iterations on the coarse grid Δ^0. The algorithm runs as follows: for the "predictor" stage, we iterate on Δ^0 to reach a tolerance $\tau_0 > 0$:

1. $\hat{u}^{n+1,0} \leftarrow \tau^0\hat{u}^n$.
2. Solve $\mathbf{J}^{n+1,m}\boldsymbol{\delta} = -\mathbf{r}^{n+1,m}$ on Δ^0 to get iterates $\mathbf{u}^{n+1,m+1} \in \mathcal{M}_1^3(\Delta^0)$. Stop when $\|\mathbf{r}^{n+1,M}\|_\infty < \tau_0$.

At this point we have a crude approximation to the new solution $\hat{u}^{n+1}$, which we use to determine the refined grid:

3. Construct Δ^{n+1} according to some refinement strategy.

Finally, we perform the "corrector" stage, iterating on Δ^{n+1} to reach a tolerance $\tau_1 > 0$:

4. $u^{n+1,M+0} \leftarrow \pi^{n+1}u^{n+1,M}$.
5. Solve $\mathbf{J}^{n+1,M+m}\boldsymbol{\delta} = -\mathbf{r}^{n+1,M+m}$ on Δ^{n+1} to get iterates $\mathbf{u}^{n+1,M+m+1} \in \mathcal{M}_1^3(\Delta^{n+1})$.
6. $\hat{u}^{n+1} \leftarrow \hat{u}^{n+1,M+m+1}$; $n \leftarrow n + 1$.

In step 5 we use the elementwise condensation algorithm outlined in the previous section to solve the linear system involving $\mathbf{J}^{n+1,M+m}$.

A sample calculation paralleling that described in Chong [10] illustrates this procedure. Consider problem (3) with N-wave data on $\Omega = (0, 1)$:

$$u_I(x) = \frac{x}{1 + \exp\left(\dfrac{4x^2-1}{8\mu}\right)}, \qquad u_0 = 0, \qquad u_N = u_I(1),$$

and let $\mu = 10^{-3}$. For the true solution, $|\partial\hat{u}/\partial x| = \mathcal{O}(1)$ except in an interior layer of thickness $\mathcal{O}(\mu)$, in which $|\partial\hat{u}/\partial x| = \mathcal{O}(\mu^{-1})$. If h denotes the coarse-grid mesh, then we insert $\mathcal{O}(\mu^{-1})$ refinement nodes in each coarse-grid element

Ω_i where $(u_i^n - u_{i-1}^n)/h > 2$. Figure 4 shows the resulting numerical solutions at different time levels, using $h = k = 0.05$, together with a plot of the exact solution for comparison.

V. IMPLEMENTATION ON A PARALLEL COMPUTER

We have implemented this refinement strategy on an Alliant FX/8 parallel processing computer. The Alliant is a shared-memory machine with optimization capability for both concurrent and vector programming. Five computational elements or processors are available on our machine as currently configured.

The computations associated with each refined coarse-grid element $\overline{\Omega}_i$ are contained in three subroutines. The first routine, called REGRID, constructs the nonsquare system of equations involving variables associated with $\overline{\Omega}_i$. The second, called CNDNS, performs the elementwise condensation and decomposition. The third, BAKSUB, solves for the refinement variables after the solution on the coarse grid is known. These routines are implemented for each refined coarse-grid element $\overline{\Omega}_i$. In each routine, calculations for separate coarse-grid elements are performed concurrently. All computation *inside* each routine must be done sequentially since the processors are in use at this time. However, the sequential calculations in each routine are optimized for vectorization.

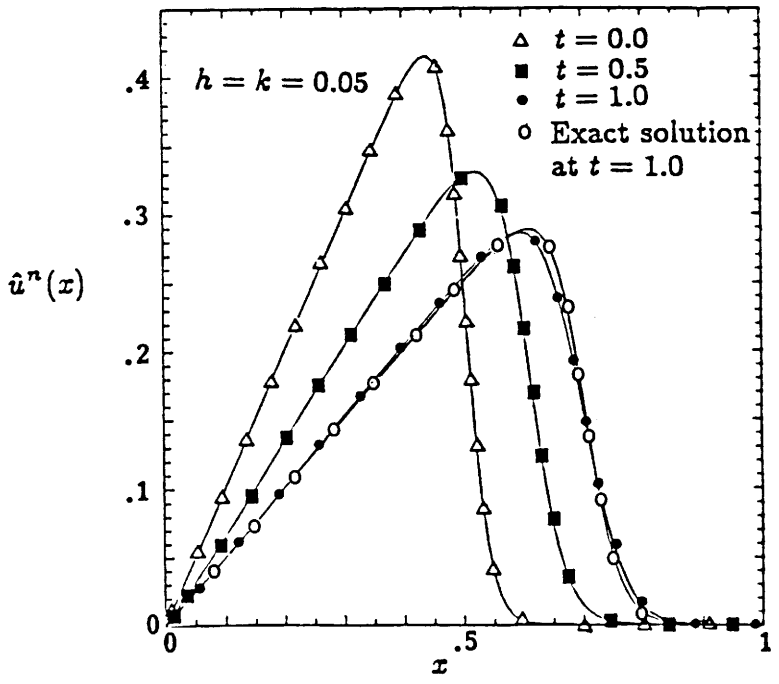


FIG. 4. Solution profiles for Burgers' equation with N-wave data, showing the exact solution for comparison in the last time step.

The machine allows users to control concurrency within a Fortran code through the use of compiler directives. The following is a description of the "corrector" stage of the nonlinear algorithm described in Section IV. The compiler directives themselves begin with the flag CVD$ starting in the first column of code.

```
        Construct the refined grid Δ^{n+1}
        Begin iteration on refined grid until ||r^{n+1,M+m+1}|| < τ_1
          Determine right hand side vector for coarse-grid equations
CVD$L     CNCALL (Compiler directive to permit the concurrent execution of the
          following loop containing a reference to an external procedure.)
          DO for each refined Ω̄_i
            CALL REGRID (Constructs nonsquare systems.)
          END DO
          Check for convergence
          Determine matrix multiplying coarse-grid variables
CVD$L     CNCALL
          DO for each refined Ω̄_i
            CALL CNDNS (Performs condensation and decomposition.)
          END DO
          Solve for coarse-grid variables
CVD$L     CNCALL
          DO for each refined Ω̄_i
            CALL BAKSUB (Solves for refinement variables.)
          END DO
        End Iteration
          .
          .
          .
CVD$R   NOCONCUR (Directive to supress concurrency until the end of the routine.)
        SUBROUTINE REGRID
CVD$R   NOCONCUR
        SUBROUTINE CNDNS
CVD$R   NOCONCUR
        SUBROUTINE BAKSUB
```

One measure of how well the algorithm makes use of the machine's parallel capabilities is the *speedup*. Speedup for n processors is the ratio of the time needed by one processor to the time used by n processors to perform the computation associated with grid refinement. If there were no overhead required to monitor and schedule the various processors, the speedup for n processors would be n. Figure 5 shows four speedup curves. These plots represent the speedups achieved by our algorithm for an average of two, four, six, and eight elements refined per time step in the Burgers' equation solver. As expected, for an average of two elements refined per time step, the speedup does not improve for more than two processors and even decreases slightly due to the increased overhead. Similarly, for an average of four elements refined in each time step, speedup does not improve when a fifth processor is used. Figure 6 shows the speedup curve when five elements are refined per time step. Clearly, this amounts to a special case for our machine configuration. The speedup for five processors is 3.51. This result compares with a machine peak of 4.5, observed by Puckett and Schmidt [11] while using a purely parallel algorithm with no

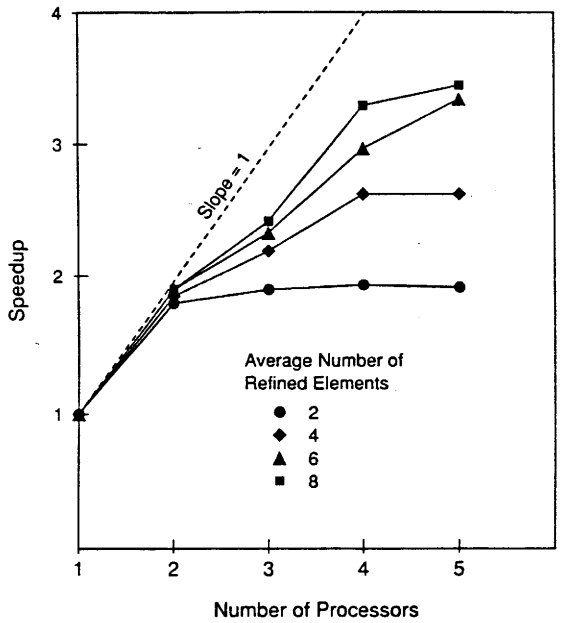


FIG. 5. Speedup plots for the parallel computations in the local gridding algorithm implemented on a five-processor machine with shared memory. Different curves represent different average numbers of coarse-grid elements refined per time step.

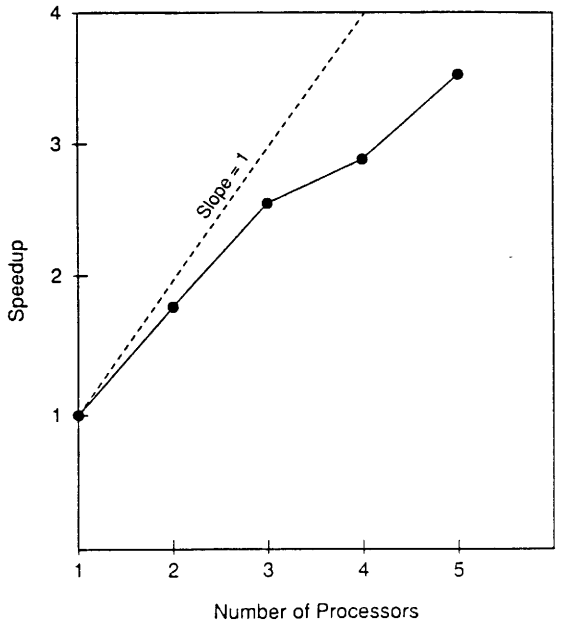


FIG. 6. Speedup plot for the parallel computations in the local gridding algorithm implemented on a five-processor machine with an average of five coarse-grid elements refined per time step.

data sharing among processors and no accumulation of results. Those authors found that peak performance occurs only when the number of iterations in a concurrent loop is quite large: They achieved the speedup of 4.5 in a loop having 3600 iterations.

There are several factors that prevent optimal speedup in our algorithm for grid refinement. First, not every processor has the same computational burden, since the amount of refinement in the coarse-grid elements can vary spatially. Second, the number of iterations performed in each loop is typically small, owing to the local nature of the refinement. A third barrier to the attainment of peak performance is the necessity to accumulate the results of the parallel computations in memory for use in subsequent calculations. These limitations seem inherent in any adaptive gridding procedure for nonlinear, transient flows. With this proviso, our algorithm appears to make good use of the shared-memory parallel architecture.

The National Science Foundation supported this work through grants DMS-8504360 and RII-8610680. The Wyoming Water Research Center also provided support through a grant-in-aid to the authors.

References

[1] B. A. Finlayson, *The Method of Weighted Residuals and Variational Principles*, Academic Press, New York, 1972.

[2] C. de Boor and B. Swartz, "Collocation at Gaussian points," *SIAM J. Numer. Anal.*, **10**, 582–606 (1973).

[3] J. Douglas and T. Dupont, "A finite element collocation method for quasi-linear parabolic equations," *Math. Comp.*, **27**, 17–28 (1973).

[4] G. Birkhoff and R. E. Lynch, *Numerical Solution of Elliptic Problems*, SIAM, Philadelphia, 1984.

[5] P. M. Prenter, *Splines and Variational Methods*, Wiley, New York, 1975.

[6] M. B. Allen and G. F. Pinder, "Collocation simulation of multiphase porous-medium flow," *Soc. Pet. Eng. J.* 135–142 (1983).

[7] O. K. Jensen and B. A. Finlayson, "Oscillation Limits for Weighted Residual Methods Applied to Convective Diffusion Equations," *Int. J. Numer. Meth. Eng.*, **15**, 1681–1689 (1980).

[8] M. B. Allen, "How upstream collocation works," *Int. J. Numer. Meth. Eng.*, **19**, 1753–1763 (1983).

[9] M. F. N. Mohsen and G. F. Pinder, "Collocation with 'adaptive' finite elements," *Int. J. Numer. Meth. Eng.*, **20**, 1901–1910 (1984).

[10] T. H. Chong, "A variable mesh finite difference method for solving a class of parabolic differential equations in one space variable," *SIAM J. Numer. Anal.* **15**, 835–857 (1978).

[11] J. A. Puckett and R. J. Schmidt, "Finite strip method in a parallel computer environment," preprint, Department of Civil Engineering, University of Wyoming, Laramie, Wyoming, 1988.

Parallel computing for solute transport models via alternating direction collocation

M. C. Curran and M. B. Allen III

Department of Mathematics, University of Wyoming, Laramie, WY 82070 U.S.A

We examine algorithmic aspects of M. Celia's alternating-direction scheme for finite-element collocation, especially as implemented for the two-dimensional advection-diffusion equation governing solute transport in groundwater. Collocation offers savings over other finite-element techniques by obviating the numerical quadrature and global matrix assembly procedures ordinarily needed in Galerkin formulations. The alternating-direction approach offers further saving in storage and serial runtime and, significantly, yields highly parallel algorithms involving the solution of problems having only one-dimensional structure. We explore this parallelism.

Key Words: Alternating-direction methods, collocation, parallel computing.

1. INTRODUCTION

Alternating-direction (AD) methods have been of interest in the numerical solution of partial differential equations since their introduction in 1955 by Peaceman and Rachford[1]. In 1970 Douglas and Dupont[2] developed an alternating-direction Galerkin method, variants of which have attracted the attention of several authors, including Dendy and Fairweather[3] and Hayes and Krishnamachari[4]. Analogous alternating-direction collocation (ADC) methods have also appeared in several papers, including those by Bangia *et al.*[5], Chang and Finlayson[6], Hayes[7], Celia *et al.*[8], Celia[9], and Celia and Pinder[10]. Reference 9, in particular, demonstrates the applicability of ADC to problems of practical importance in water resources engineering.

We examine Celia's ADC for the two-dimensional advection-diffusion equation for solute transport in a known velocity field. Of interest here are algorithmic features of ADC that enhance its efficiency in comparison with standard two-dimensional collocation, especially the amenability of ADC to implementation on parallel-architecture computers. The paper has the following structure: section 2 briefly reviews finite-element collocation using bicubic Hermite bases; section 3 discusses the AD method applied to collocation; in section 4 we discuss the method's performance on a parallel computer.

2. REVIEW OF FINITE-ELEMENT COLLOCATION

We begin by reviewing finite-element collocation for problems in two space dimensions. The primary aim of this review is to establish notation and terminology for the rest of the paper. Lapidus and Pinder[11] give an alternative, more detailed description of the methodology that may be more appropriate for those seeking an introduction.

Consider the following problem, posed on the rectangular spatial domain $\Omega = (a, b) \times (c, d)$:

$$\text{(a)}\ \partial_t u + \mathbf{v}\cdot\nabla u - \nabla\cdot(D\nabla u) = 0, (x, y, t) \in \Omega \times (0, \infty),$$

$$\text{(b)}\ u(x, y, 0) = u_I(x, y),\ (x, y) \in \Omega, \tag{1}$$

$$\text{(c)}\ u(x, y, t) = u_B(x, y, t),\ (x, y) \in \partial\Omega,\ t \geqslant 0.$$

In equation (1a), $\mathbf{v} = \mathbf{v}(x, y)$ represents a known fluid velocity, which in applications might be the Darcy velocity computed using a groundwater flow model. $D = D(x, y)$ is a diffusion coefficient, which in underground flows could serve as a simple model of hydrodynamic dispersion. (For purposes of testing the efficiency of collocation algorithms, we neglect the possible tensorial nature of D and suppress explicit consideration of any dependence on the fixed velocity field $\mathbf{v}$.) The unknown function $u = u(x, y, t)$ represents a solute concentration. Equation (1b) gives the initial concentration field, while equation (1c) imposes Dirichlet boundary conditions. These boundary data are not the only ones to which the ADC method applies; in fact, one could just as well impose Neumann, Robin, or mixed boundary conditions.

We use finite-element collocation to discretize the spatial dimensions in the following class of semidiscrete analogs:

$$u^{n+1} - u^n + k[\mathbf{v}\cdot\nabla u^{n+\theta} - \nabla\cdot(D\nabla u^{n+\theta})] = 0,$$
$$n = 0, 1, 2, \ldots, \tag{2}$$

where integer superscripts indicate time level. The notation $(\cdot)^{n+\theta}$ signifies a convex combination $\theta(\cdot)^{n+1} + (1-\theta)(\cdot)^n$ of the quantity $(\cdot)$ at successive time levels, where $0 \leqslant \theta \leqslant 1$, and k denotes the time step. In particular, the choice $\theta = 1/2$ yields a Crank-Nicolson scheme, for which we expect the local truncation error to be $O(k^2)$.

Accepted September 1989. Discussion closes December 1990.

We begin by establishing a rectangular grid on Ω and a corresponding space of finite-element interpolating functions. Let $\Delta_x = \{a = x_0, \ldots, x_{N_x} = b\}$ and $\Delta_y = \{c = y_0, \ldots, y_{N_y} = d\}$ be grids on the x- and y-intervals (a, b) and (c, d), respectively, and call $h_x = \max_{1 \leqslant i \leqslant N_x} \{x_i - x_{i-1}\}$ and $h_y = \max_{1 \leqslant j \leqslant N_y} \{y_j - y_{j-1}\}$. The Hermite piecewise cubics on these one-dimensional grids are functions belonging to the spaces

$$\mathcal{M}_1^3(\Delta_x) = \{f \in C^1([a, b]) \mid f|_{[x_{i-1}, x_i]} \text{ is cubic, } i = 1, \ldots, N_x\},$$

$$\mathcal{M}_1^3(\Delta_y) = \{f \in C^1([c, d]) \mid f|_{[y_{j-1}, y_j]} \text{ is cubic, } j = 1, \ldots, N_y\},$$

Here $f|_{[x_{i-1}, x_i]}$ denotes the restriction of the globally defined function f to the subinterval $[x_{i-1}, x_i]$. Thus each function in either of these spaces agrees with some cubic polynomial on any subinterval in the grid, and these cubic 'pieces' connect in a manner that preserves global continuous differentiability. As Prenter[12] shows, each of these spaces has an interpolating basis $\{h_{0i}, h_{1i}\}_{i=0}^{N_x \text{ or } N_y}$, every element of which has support confined to at most two adjacent subintervals $[x_{i-1}, x_i]$ or $[y_{j-1}, y_j]$. Given any function $f \in \mathcal{M}_1^3(\Delta_x)$, for example, the representation of f with respect to this basis takes the form

$$f(x) = \sum_{i=0}^{N_x} [f(x_i)h_{0i}(x) + f'(x_i)h_{1i}(x)].$$

For the two-dimensional problem (1), we use these interpolating spaces to form a tensor-product interpolating space $\mathcal{M}_1^3(\Delta_x) \otimes \mathcal{M}_1^3(\Delta_y)$. This space has a basis in which each function is the product of a piecewise cubic basis function in $\mathcal{M}_1^3(\Delta_x)$ and one in $\mathcal{M}_1^3(\Delta_y)$. At each time level n, we compute an approximate solution $\hat{u}^n(x, y)$ belonging to the trial space

$$\mathcal{M} = \{v \in \mathcal{M}_1^3(\Delta_x) \otimes \mathcal{M}_1^3(\Delta_y) \mid v(x, y) = u_B(x, y) \; \forall \, (x, y) \in \partial\Omega\}.$$

As the notation indicates, each function in $\mathcal{M}$ automatically obeys the boundary conditions (1c) and has the form

$$\begin{aligned}\hat{u}^n(x, y) = \sum_{i=0}^{N_x} \sum_{j=0}^{N_y} [&\hat{u}^n(x_i, y_j)H_{00ij}(x, y) \\ &+ \partial_x\hat{u}^n(x_i, y_j)H_{10ij}(x, y) \\ &+ \partial_y\hat{u}^n(x_i, y_j)H_{01ij}(x, y) \\ &+ \partial_{xy}\hat{u}^n(x_i, y_j)H_{11ij}(x, y)],\end{aligned}$$

where $H_{lmij}(x, y) = h_{li}(x)h_{mj}(y)$.

At $t = 0$ we form the initial approximate solution $\hat{u}^0$ by using the nodal values of the initial function u_I and its x-, y-, and xy-derivatives to form the projection of the true initial concentration onto $\mathcal{M}$. These criteria specify $\hat{u}^0$ completely. For subsequent time levels, the fact that every function in the trial space $\mathcal{M}$ satisfies the boundary conditions fixes the nodal values and tangential derivatives of the approximate solute concentration along the boundary $\partial\Omega$. A careful count will reveal that the boundary conditions determine $4(N_x + N_y + 1)$ of the $4(N_x + 1)(N_y + 1)$ nodal coefficients for each unknown function $\hat{u}^1, \hat{u}^2, \ldots$.

At each new time level $n + 1$, we use our knowledge of the most recently computed approximate solution $\hat{u}^n$ to determine the remaining $4N_xN_y$ degrees of freedom for $\hat{u}^{n+1}$. We first form the residual

$$R^{n+1} = \hat{u}^{n+1} - \hat{u}^n + k[\mathbf{v} \cdot \nabla\hat{u}^{n+\theta} - \nabla \cdot (D\nabla\hat{u}^{n+\theta})].$$

We then pick a collection $\{(\bar{x}_1, \bar{y}_1), (\bar{x}_1, \bar{y}_2), \ldots, (\bar{x}_{2N_x}, \bar{y}_{2N_y})\}$ of $4N_xN_y$ collocation points and force $R^{n+1}(\bar{x}_p, \bar{y}_q) = 0$ at each, thus enforcing precisely the correct number of conditions to determine $\hat{u}^{n+1}$. In particular, we choose $\bar{x}_p$ and $\bar{y}_q$ to be the two-point Gauss-quadrature abscissae on each subinterval $[x_{i-1}, x_i]$ or $[y_{j-1}, y_j]$. Since the spatial problem to be solved at each time level is elliptic we expect this choice of collocation points to yield optimal global error estimates of the form $\|u^n - \hat{u}^n\|_\infty = \mathcal{O}(h_x^4 + h_y^4)$ (see Refs 13 and 14).

3. THE ALTERNATING-DIRECTION METHOD

The aim of ADC is to modify the ordinary two-dimensional collocation procedure via an operator splitting. This splitting reduces the discrete problem to one involving a sequence of matrix equations, each of which has the same sparse structure as the one-dimensional collocation system. The following description of this splitting approach is essentially a review of the development presented by Celia and Pinder in Ref. 10.

We first perturb equation (2) by a term that is $\mathcal{O}(k^2)$ to get

$$u^{n+1} - u^n + k(\mathcal{L}_x + \mathcal{L}_y)u^{n+\theta} + k^2\theta^2(\mathcal{L}_x\mathcal{L}_y)(u^{n+1} - u^n) = 0, \quad (3)$$

where

$$\mathcal{L}_x = v_x\partial_x - \partial_x(D\partial_x); \qquad \mathcal{L}_y = v_y\partial_y - \partial_y(D\partial_y).$$

(Reference 10 treats the advection-diffusion equation in a slightly different fashion, splitting only the diffusive part of the spatial operator.) Rearranging equation (3) and factoring gives

$$(1 + k\theta\mathcal{L}_y)(1 + k\theta\mathcal{L}_x)(u^{n+1} - u^n) = -k(\mathcal{L}_x + \mathcal{L}_y)u^n.$$

Conceptually, we can solve $(1 + k\theta\mathcal{L}_y)z = -k(\mathcal{L}_x + \mathcal{L}_y)u^n$ for the intermediate unknown z, then solve $(1 + k\theta\mathcal{L}_x)(u^{n+1} - u^n) = z$ for the time increment in $\hat{u}$.

To see how this works algebraically, notice that substituting Hermite bicubic trial functions for $\hat{u}$ and collocating produces a matrix equation $\mathbf{K}\mathbf{u}^{n+1} = \mathbf{r}^n$, where $\mathbf{u}^{n+1}$ is the vector of time increments for the unknown nodal coefficients of $\hat{u}^{n+1}$. Consider a typical entry of the matrix $\mathbf{K}$:

$$\{[1 + k\theta(\mathcal{L}_x + \mathcal{L}_y) + k^2\theta^2(\mathcal{L}_x\mathcal{L}_y)]H_{lmij}\}(\bar{x}_p, \bar{y}_q), \quad (4)$$

where H_{lmij} is some basis function in the tensor-product interpolation space. Each $H_{lmij}(x, y) = h_{li}(x)h_{mj}(y)$, so we can expand the expression (4) and factor it to get

$$[h_{li}(\bar{x}_p) + k\theta(\mathcal{L}_x h_{li})(\bar{x}_p)] \cdot [h_{mj}(\bar{y}_q) + k\theta(\mathcal{L}_y h_{mj})(\bar{y}_q)].$$

This factoring of each matrix entry, together with Celia's scheme[9] for numbering and renumbering equations and unknowns, allows us to factor the entire matrix equation at each time level in a computationally attractive fashion. If we number the equations and unknowns 'vertically,' that is, consecutively along the lines $x = \bar{x}_p$, as shown in Fig. 1a, then the

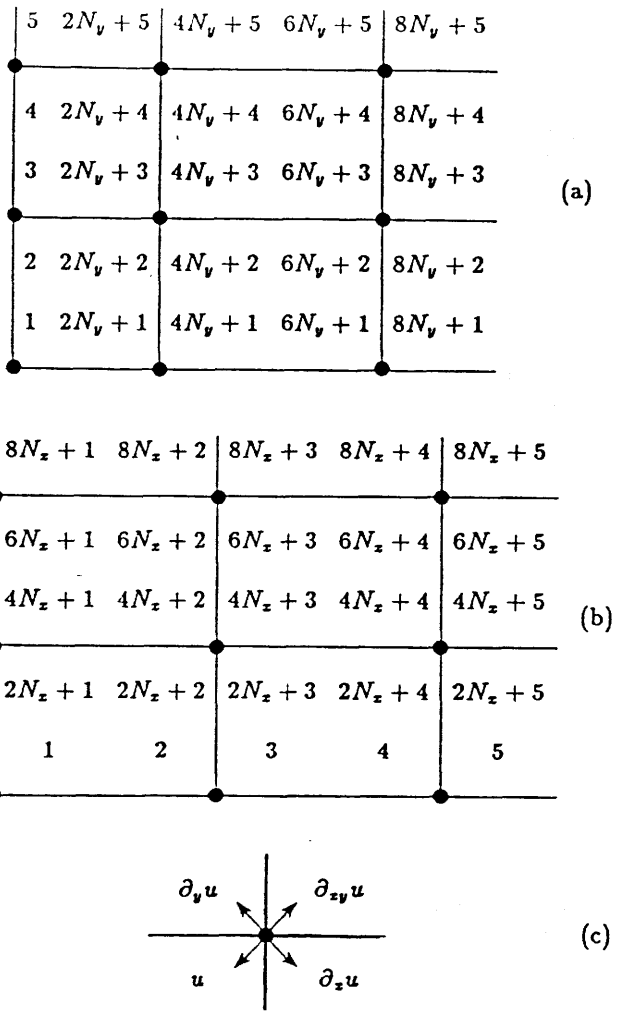


Fig. 1. (a) Vertical numbering scheme for the equations used in the y-sweep. Equation numbers occupy the sites of corresponding collocation points; the symbols • indicate nodes in the grid. (b) Horizontal numbering scheme for the equations used in the x-sweep. (c) Association scheme for numbering nodal unknowns following a given numbering scheme for the collocation points surrounding the node.

$4N_xN_y \times 4N_xN_y$ matrix $\mathbf{K}$ factors as follows:

$$\mathbf{K} = \mathbf{YX} = \begin{bmatrix} \mathbf{Y}_{1,1} & & \\ & \ddots & \\ & & \mathbf{Y}_{2N_x,2N_x} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,1} & \cdots & \mathbf{X}_{1,2N_y} \\ \vdots & & \vdots \\ \mathbf{X}_{2N_y,1} & \cdots & \mathbf{X}_{2N_y,2N_y} \end{bmatrix}.$$

Each $2N_y \times 2N_y$ block $\mathbf{Y}_{p,p}$ has the five-band structure of a one-dimensional collocation matrix, shown in Fig. 2. Moreover, The entries in $\mathbf{Y}_{p,p}$ depend only on the y-coordinates of collocation points.

Now consider the matrix $\mathbf{X}$. If we switch to the 'horizontal' numbering scheme for equations and unknowns, illustrated in Fig. 1b, then $\mathbf{X}$ transforms to a block-diagonal matrix that we denote as follows:

$$\mathbf{X}^* = \begin{bmatrix} \mathbf{X}^*_{1,1} & & \\ & \ddots & \\ & & \mathbf{X}^*_{2N_y,2N_y} \end{bmatrix}$$

(We use the superscript * to indicate the result of switching to the 'horizontal' numbering scheme.) Again, each $2N_x \times 2N_x$ block $\mathbf{X}^*_{q,q}$ has the five-band structure shown in Fig. 2.

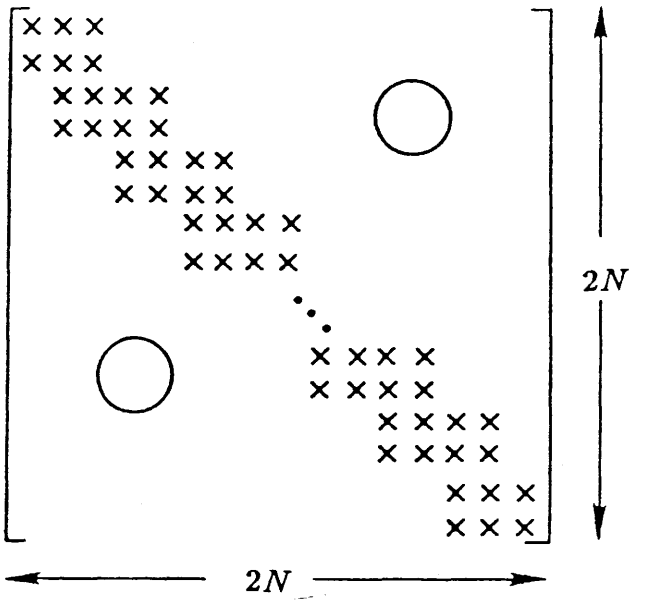


Fig. 2. Five-band zero structure associated with the matrix for standard one-dimensional collocation

In light of these observations, we can solve the two-dimensional matrix equation $\mathbf{Ku}^{n+1} = \mathbf{r}^n$ by the following procedure.

1. Adopt the 'vertical' numbering scheme, and solve $\mathbf{Yz} = \mathbf{r}^n$ for the intermediate vector $\mathbf{z}$ by solving the independent problems $\mathbf{Y}_{p,p}\mathbf{z}_p = \mathbf{r}^n_p$, $p = 1, \ldots, 2N_x$.
2. Renumber according to the 'horizontal' scheme, converting $\mathbf{z}$ to the reordered vector $\mathbf{z}^*$. This renumbering transforms $\mathbf{X}$ to the block-diagonal form $\mathbf{X}^*$.
3. Solve $\mathbf{X}^*\mathbf{u}^{n+1} = \mathbf{z}^*$ for the desired time increments by solving the independent systems $\mathbf{X}^*_{q,q}\mathbf{u}^{n+1}_q = \mathbf{z}^*_q$, $q = 1, \ldots, 2N_y$.

Thus each time step involves the solution of matrix equations that are at worst one-dimensional in structure.

At this point we can make some comments regarding the efficiency to be gained by the splitting scheme. For simplicity, let us assume that $N_x = N_y = N$. In the fully two-dimensional matrix problem $\mathbf{Ku}^{n+1} = \mathbf{r}^n$, there are then $4N^2$ unknowns, and the matrix $\mathbf{K}$ is asymmetric. If we order equations and unknowns to allow for row reduction without pivoting, $\mathbf{K}$ will have a bandwidth $B_2 = 8N + 16$ (see Ref. 15). Assuming that row reduction accounts for the bulk of the computational work in the sparse matrix solver used, we can expect the operation count for solving the fully two-dimensional equations at each time step to be roughly $4N^2B_2^2 = 256N^4$ for large N. By contrast, ADC calls for the solution of $4N$ matrix equations of bandwidth $B_1 = 5$ and order $2N$ at each time level. Thus an upper bound for the number of arithmetic operations required in the row reductions for ADC is $4N(2NB_1^2) = 200N^2$.

Furthermore, each of the 'one-dimensional' systems in steps 1 and 3 of ADC is independent of any other.

Therefore these steps can run concurrently, whereas there appears to be no such obvious parallelism in the standard solvers for the fully two-dimensional formulation. We explore the inherent parallelism of ADC in the next section.

4. IMPLEMENTATION ON A PARALLEL COMPUTER

We have implemented ADC on an Alliant FX/8 parallel processing computer. The Alliant has eight processors in a shared-memory configuration in which each processor is a vector-architecture machine. The Alliant allows users to control concurrency within a standard Fortran code through the use of compiler derectives. Since we are mainly interested in the general advantages to be gained through the shared-memory architecture and the concurrency controls furnished by the compiler, we shall not consider such other machine-specific features as size of the cache (high-speed memory), number of processors, or speed of the random-access memory.

The following is a description of the code outlined in Steps 1–3 of section 3. The compiler directives themselves begin with the flag CVD$ starting in the first column of code.

```
        Initialize û⁰, set n = 0
        Begin time level n + 1
CVD$L     CNCALL (Compiler directive to permit the con-
                  current execution of the following loop
                  containing a reference to an external
                  procedure.)
          DO for each p = 1, ..., 2N_x
            CALL YSWEEP (Constructs the system
                         Y_{p,p} z_p = r_p^n, solves it, and
                         saves the results.)
          END DO
          CALL RENUM (Reorders z to get z*)
CVD$L     CNCALL
          DO for each q = 1, ..., 2N_y
            CALL XSWEEP (Constructs the system
                         X*_{q,q} u_q^{n+1} = z*_q, solves it and
                         updates the nodal coefficients
                         of û to time level n + 1.)
          END DO
        End time step
          ⋮
CVD$R NOCONCUR (Directive to supress concurrency
                until the end of the subroutine.)
        SUBROUTINE YSWEEP
CVD$R NOCONCUR
        SUBROUTINE XSWEEP
```

The directive CNCALL forces the passes through a DO loop to execute in parallel, within the limitations of the machine's configuration. Thus, for example, if the algorithm calls for eight passes through the loop and there are eight processors, then CNCALL forces the operating system to map each pass onto a separate processor, allowing concurrent execution of the passes. If, on the same machine, the algortithm calls for nine passes through the loop, then the last pass must wait until one of the first eight terminates before the operating system can map the ninth onto a free processor. This logic implies that certain efficiencies accrue when the number of independent processes is an integer multiple of the number of processors in the machine being used.

The need for the directive NOCONCUR arises from the structure of the Alliant's optimizing compiler, which often must choose among several levels of parallelism in a code. By default, the compiler optimizes for parallelism at the finest level. Thus, for example, it will force independent processes *within* a subroutine to run concurrently, in preference to forcing independent calls to the subroutine itself to run concurrently. Inserting the directive NOCONCUR before the SUBROUTINE statement overrides the default level for optimization. This device allows the compiler to treat each call to the subroutine as an independent process, even if the potential for concurrency exists at a finer level inside the subroutine.

One measure of how well the algorithm makes use of the machine's parallel capabilities is the *speedup*. Speedup for n processors is the ratio of the CPU time needed by one processor to the time used by n processors to perform a set of tasks in parallel. For a perfectly parallel algorithm requiring no overhead to monitor or schedule the various processes and no storage of their results, the speedup for n processors would be n. Figure 3 shows the speedup curve for the ADC algorithm, implemented for the advection-diffusion problem on a 40×40-element spatial grid. The CPU time used to compute these ratios is actual clock time, excluding the processing required for initializing the code but including computational overhead required for scheduling and storage of intermediate results. The speedup curve is quite close to the ideal curve of unit slope, yielding a speedup of 7.27 for eight processors. Clearly, ADC makes very good use of the Alliant's shared-memory parallel architecture.

We caution against extrapolating these speedup results to much larger problems on the Alliant as configured. The size of the cache memory in any particular

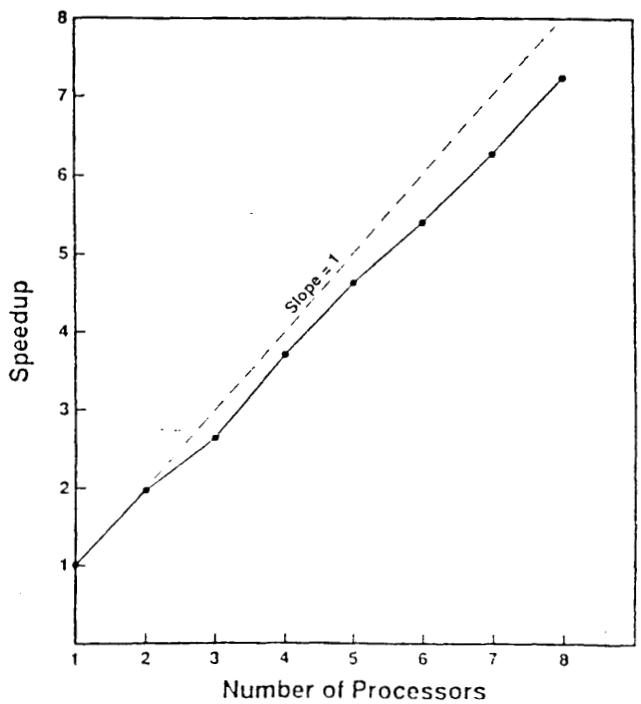


Fig. 3. Speedup curve for ADC using the Alliant FX/8 shared-memory architecture

computer clearly constrains that machine's ability to compute efficiently. What is important here is not the computational horsepower of the particular machine we have used but rather the natural parallelism inherent in the ADC algorithm. This parallelism can yield significant speedups on essentially any shared-memory parallel architecture.

To confirm that ADC gives useful approximations, Figs 4 and 5 show solution plots for two different problems. Figure 4 shows the results of a rotating plume problem on $\Omega = (-1, 1) \times (-1, 1)$, with $N_x = N_y = 40$ and $k = 0.004$. Here, $\mathbf{v} = 2\pi(-y, x)$ is a circular velocity field, $D = 0$, and the initial concentration plume $u_I(x, y)$ is a 'Gauss hill' with center at $(0, -0.6)$ and standard deviation $\sigma = 0.066$. This pure advection problem, while physically unrealistic, poses a fairly severe test of ADC's ability to approximate solutions with steep fronts in highly advective flow fields. In this case, the global error at $t = 1$, when the exact solution is identical to the initial condition, is less than $0.08 \| u \|_\infty$.

Figure 5 displays the results of an advection-diffusion problem on $\Omega = (0, 1) \times (0, 1)$, with $N_x = N_y = 20$ and $k = 0.004$. The diffusion coefficient here is $D = 0.00385$. The velocity field is $\mathbf{v}(x, y) = 2e^{xy}(x, -y)$, which corresponds to the steady-state Darcy velocity $-K\nabla\Phi$ on Ω when the hydraulic conductivity is $K(x, y) = e^{xy}$ and the head obeys the boundary conditions $\Phi(x, y) = x^2 - y^2$ on $\partial\Omega$. The inital concentration distribution u_I for this problem is another 'Gauss hill,' with $\sigma = 0.05$ and center $(0.75, 0.25)$.

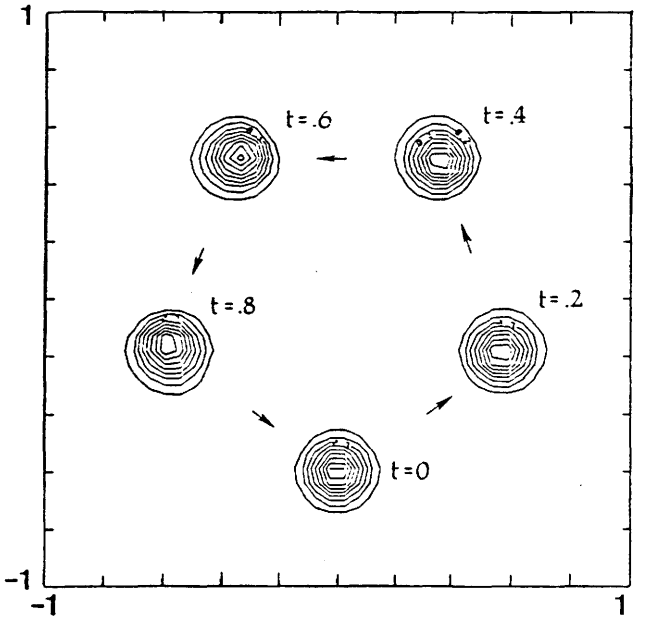


Fig. 4. Concentration contours for the purely advective rotating plume problem at various time levels. Contour interval is 0.1

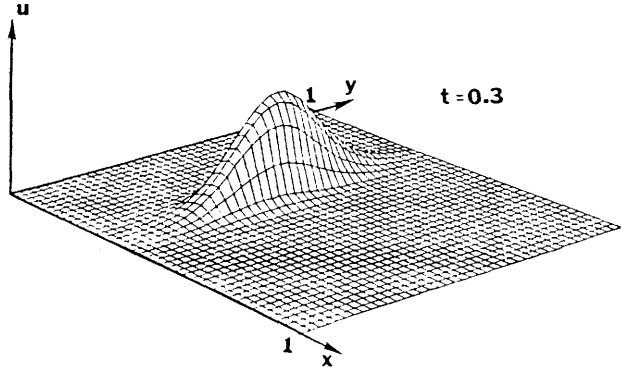


Fig. 5. Plot of concentration distribution at $t = 0.3$ for an advection-diffusion problem with Darcy flow

5. CONCLUSIONS

From operation counts alone, it has been clear for some time that ADC offers distinct efficiencies over standard methods for two-dimensional collocation in a serial computing environment. With the advent of practical parallel computers, ADC holds even more promise, since the splitting scheme converts a fully two-dimensional problem into a sequence of 'one-dimensional' problems that are amenable to concurrent processing. Similar observations should hold for other alternating-direction methods, including techniques for multidimensional finite-difference and Galerkin aproximations.

ACKNOWLEDGMENTS

The Wyoming Water Research Center supported this work. We also received support from NSF grant RII-8610680, EPA cooperative agreement CR813928-01-0, and ONR contract 0014-88-K-0370.

REFERENCES

1 Peaceman, D. W. and Rachford, H. H. The numerical solution of parabolic and elliptic equations, *SIAM J.* 1955, **3**, 28–41

2 Douglas, Jr., J. and Dupont, T. Alternating-direction Galerkin methods on rectangles, *Numerical Solution of Partial Differential Equations, Vol. 2*, B. Hubbard, ed., Academic, New York, 1971, 133–214

3 Dendy, J. and Fairweather, G. Alternating-direction Galerkin schemes for parabolic and hyperbolic problems on rectangular polygons, *SIAM J. Numer. Anal.* 1975, **2**, 144–163

4 Hayes, L. J. and Krishnamachari, S. V. Alternating direction along flow lines in a fluid flow problem, *Comp. Meth. App. Mech. and Engg* 1989, **47**, 187–203

5 Bangia, V. K., Bennett, C. and Reynolds, A. Alternating direction collocation for simulating reservoir performance, 53rd annual fall conference, Society of Petroleum Engineers, Houston, 1978

6 Chang, P. W. and Finlayson, B. A. Orthogonal collocation on finite elements for elliptic equations, *Math. Comp. Simulation*, 1978, 83–92

7 Hayes, L. J. An alternating-direction collocation method for finite element approximations on rectangles, *Comput. Math. Appl.*, 1980, **6**, 45–50

8 Celia, M. A., Pinder G. F. and Hayes, L. J. Alternating direction collocation simulation of the transport equation, *Proceedings Third Int. Conf. Finite Elements in Water Resources*, S. Y. Wang *et al.*, eds., University of Mississippi, Oxford, MS, 1980, 3.36–3.48

9 Celia, M. A., *Collocation on deformed finite elements and alternating direction collocation methods*, Ph.D. Dissertation, Princeton University, 1983

10 Celia, M. A. and Pinder, G. F. Analysis of alternating-direction methods for parabolic equations, *Numer. Meth. P.D.E.* 1985, **1**, 57–70

11 Lapidus, L. and Pinder, G. F. *Numerical solution of partial differential equations in science and engineering*, New York, 1982
12 Prenter, P. M. *Splines and variational methods*, New York, 1975
13 Percell, P. and Wheeler, M. F., A C^1 finite element collocation method for elliptic problems, *SIAM J. Numer. Anal. 17* 1980, 605–622
14 Dyksen, W. R., Lynch, R. E., Rice, J. R. and Houstis, E. N. The performance of the collocation and Galerkin methods with hermit bicubics, *SIAM J. Numer. Anal.*, 1984, **21**, 675–715
15 Frind, E. O. and Pinder, G. F. A collocation finite element method for potential problems in irregular domains, *Int. J. Numer. Meth. Engg* 1979, **14**, 681–701

WELL CONDITIONED ITERATIVE SCHEMES FOR MIXED FINITE-ELEMENT MODELS OF POROUS-MEDIA FLOWS

Myron B. Allen and Richard E. Ewing
University of Wyoming

Peng Lu
University of Georgia

Key Words: Mixed finite-elements, iterative solution schemes, heterogeneous porous media. AMS (MOS) subject classification: 65, numerical analysis.

Abstract

Mixed finite-element methods are attractive in modeling flows in porous media, since they can yield pressures and velocities having comparable accuracy. In solving the resulting discrete equations, however, poor matrix conditioning can arise both from spatial heterogeneity in the medium and from the fine grids needed to resolve that heterogeneity. This paper presents iterative schemes that overcome these sources of poor conditioning by using effective preconditioners in conjunction with a multigrid method for pressures.

1 Introduction

We consider methods for solving discrete approximations to the equations governing single-fluid flow in a porous medium. If the flow is steady and two-dimensional with no gravity drive, Darcy's law and the mass balance take the following forms:

$$\begin{aligned} \mathbf{u} &= -K \operatorname{grad} p \quad \text{in } \Omega, \\ -\ \operatorname{div} \mathbf{u} &= f \quad \text{in } \Omega. \end{aligned} \tag{1.1}$$

Here $\mathbf{u}, p, f$ represent the Darcy velocity, pressure, and source term, respectively. For simplicity, we take the spatial domain to be a square, scaled so that $\Omega = (0,1) \times (0,1)$. The coefficient $K(x,y)$ is the *mobility*, defined as the ratio of the permeability of the porous medium to the dynamic viscosity of the fluid. In applications to underground flows, the structure of K may be quite complex, depending on the lithology of the porous medium

and the composition of the fluid. We assume, however, that this ratio is bounded and integrable on $\bar{\Omega}$ and satisfies $K \geq K_{\text{inf}} > 0$. We impose the boundary condition $p = 0$ on $\partial\Omega$, so that p effectively represents the deviation in pressure from a reference value known along $\partial\Omega$.

Scientists modeling contaminant flows in groundwater or solvent flows in oil reservoirs often need accurate finite-element approximations of $\mathbf{u}$ and p simultaneously. For this reason, mixed finite-element methods for solving the system (1.1) are particularly attractive, since they can yield approximations to $\mathbf{u}$ and p that have comparable accuracy ([1], [5]). The key to achieving such approximations is the use of appropriate piecewise polynomial trial spaces, such as those proposed by Raviart and Thomas [10]. As we review in Section 2, if we use the lowest-degree Raviart-Thomas spaces, the mixed formulation yields systems of discrete equations that have the form

$$\begin{aligned} AU + NP &= 0, \\ N^T U &= F. \end{aligned} \tag{1.2}$$

Here, U and P signify vectors containing nodal values of the trial functions for $\mathbf{u}$ and p, defined on a grid over Ω, and A and N are matrices. As we illustrate below, the matrix A contains all information about the spatially varying material property K, while N and N^T are essentially finite-difference matrices.

Equations (1.2) can be quite difficult to solve efficiently, for the following reasons. When K varies over short distances, accurate finite-element approximations require fine grids on Ω. Fine grids, however, typically yield poorly conditioned matrix equations. For classical stationary iterative schemes, this increase in the condition number of the system leads to slow convergence, no matter how "nice" K may be ([2], Section 4.11). The problem is compounded whenever K exhibits large spatial variations, as can occur near lithologic changes in the porous medium or sharp contacts between fluids of different viscosity. In such problems, as we shall demonstrate, the poor conditioning associated with spatial variability typically aggravates that associated with the fine grids needed to resolve the physics of the problem. Thus, in problems with significant material heterogeneity, methods that are relatively insensitive to these two sources of poor conditioning can have considerable utility.

In this paper we discuss two types of iterative schemes for the mixed-method equations

(1.2). The first type possesses convergence rates that are independent of the fineness of the grid. The second type, derived from the first, also overcomes the sensitivity to the spatial structure of K, at the expense of somewhat more computation per iteration. Briefly, the first scheme proceeds as follows: Let $(U^{(0)}, P^{(0)})$ be initial guesses for the value of (U, P). Then the k-th iterate for (U, P) is the solution of

$$\begin{pmatrix} \omega I & N \\ N^T & 0 \end{pmatrix} \begin{pmatrix} U^{(k)} \\ P^{(k)} \end{pmatrix} = \begin{pmatrix} 0 \\ -F \end{pmatrix} + \begin{pmatrix} \omega I - A & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U^{(k-1)} \\ P^{(k-1)} \end{pmatrix}, \tag{1.3}$$

where I stands for the identity matrix and ω signifies a parameter, discussed below, that is related to the spectral radius $\rho(A)$ of A. For each iteration level k, the main computational work in (1.3) is to solve a linear system of the form $(\omega^{-1}N^TN)P^{(k)} = G^{(k-1)}$. However, the matrix $\omega^{-1}N^TN$ remains vulnerable to the poor conditioning associated with fine grids. We overcome this difficulty by using a multigrid scheme to solve for $P^{(k)}$, thereby greatly reducing the computational work in each iteration.

An interesting feature of this approach is that N^TN is essentially the matrix associated with the five-point difference approximation to the Laplace operator with Dirichlet boundary conditions. Hence, the multigrid portion of the scheme does not encounter the variable coefficient, and the algorithm is particularly simple. The price paid for this simplicity, as we shall see, is sensitivity to the poor conditioning associated with spatial variability.

To overcome this second source of trouble, we modify the first scheme to get new ones of the form

$$\begin{pmatrix} D & N \\ N^T & 0 \end{pmatrix} \begin{pmatrix} U^{(k)} \\ P^{(k)} \end{pmatrix} = \begin{pmatrix} 0 \\ -F \end{pmatrix} + \begin{pmatrix} D - A & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U^{(k-1)} \\ P^{(k-1)} \end{pmatrix}, \tag{1.4}$$

where D denotes a diagonal matrix that we compute from A. This new class of schemes requires us to invert N^TDN, which we again do using a multigrid method to preserve h-independence of the convergence rate. While the multigrid method must now accommodate spatially varying coefficients, the overall scheme possesses the advantage that its convergence rate is essentially independent of the spatial structure of K.

Our paper has the following format: In section 2 we review the mixed finite-element method that we use. Section 3 describes the first iterative scheme in more detail and analyzes its convergence. In section 4 we discuss the application of multigrid ideas to the first scheme, and in section 5 we present some numerical results for this algorithm. Section

6 describes the modifications necessary to produce the second class of iterative schemes and presents numerical results illustrating good convergence rates even in the presence of heterogeneities.

2 A Mixed Finite-Element Method

We begin with a brief review of the mixed finite-element method, following the notation of Ewing and Wheeler [8]. Let $H(\mathrm{div},\Omega) = \{\mathbf{v} \in L^2(\Omega) \times L^2(\Omega) : \mathrm{div}\ \mathbf{v} \in L^2(\Omega)\}$. The variational form for (1.1) is as follows: Find a pair $(\mathbf{u}, p) \in H(\mathrm{div},\Omega) \times L^2(\Omega)$ such that

$$\begin{aligned} \int_\Omega \frac{\mathbf{u}\cdot\mathbf{v}}{K} dxdy + \int_\Omega p\ \mathrm{div}\ \mathbf{v}\, dxdy &= 0, \quad \forall\, \mathbf{v} \in H(\mathrm{div},\Omega), \\ \int_\Omega (\mathrm{div}\ \mathbf{u} + f) q\, dxdy &= 0, \quad \forall\, q \in L^2(\Omega). \end{aligned} \tag{2.1}$$

By our assumptions on K, there exist constants $K_{\mathrm{inf}}, K_{\mathrm{sup}}$ such that $0 < K_{\mathrm{inf}} \le K \le K_{\mathrm{sup}}$. Implicit in these equations is also the assumption that K^{-1} is integrable on $\bar{\Omega}$.

To discretize the system (2.1), let $\Delta_x = \{0 = x_0 < x_1 < \cdots < x_m = 1\}$ be a set of points on the x-axis and $\Delta_y = \{0 = y_0 < y_1 < \cdots < y_n = 1\}$ a set of points on y-axis. Let $\Delta_h = \Delta_x \times \Delta_y$ be the rectangular grid on Ω with nodes $\{(x_i, y_j)\}_{i=0,j=0}^{m,n}$. The mesh of this grid is

$$h = \max_{i,j}\{x_i - x_{i-1},\, y_j - y_{j-1}\}.$$

We assume throughout the paper that Δ_x and Δ_y are quasiuniform in the sense that $x_i - x_{i-1} \ge \alpha h$ and $y_j - y_{j-1} \ge \alpha h$ for some fixed $\alpha \in (0,1)$. With Δ_h we associate a finite-element subspace $\mathbf{Q}_h \times V_h$ of $H(\mathrm{div},\Omega) \times L^2(\Omega)$. The "velocity space" is $\mathbf{Q}_h = Q_h^x \times Q_h^y$, where Q_h^x and Q_h^y are both tensor-product spaces of one-dimensional finite-element spaces. In particular, we use the lowest-order Raviart-Thomas spaces, in which Q_h^x contains functions that are piecewise linear and continuous on Δ_x and piecewise constant on Δ_y. Similarly, Q_h^y contains functions that are piecewise linear and continuous on Δ_y and piecewise constant on Δ_x. The "pressure space" V_h consists of functions that are piecewise constant on Δ_h.

Given these approximating spaces, the corresponding mixed finite-element method for

solving Equations (2.1) is as follows: Find a pair $(\mathbf{u}_h, p_h) \in \mathbf{Q}_h \times V_h$ such that

$$\begin{aligned} \int_\Omega \frac{\mathbf{u}_h \cdot \mathbf{v}_h}{K} dxdy + \int_\Omega p_h \text{ div } \mathbf{v}_h dxdy = 0, \quad \forall\, \mathbf{v}_h \in \mathbf{Q}_h, \\ \int_\Omega (\text{div } \mathbf{u}_h + f) q_h dxdy = 0, \qquad\qquad \forall\, q_h \in V_h. \end{aligned} \tag{2.2}$$

This finite-element discretization yields approximations $\mathbf{u}_h$ and p_h whose global errors are both $O(h)$ in the norm $\|\cdot\|_{L^2(\Omega)}$. Ewing et al. [6] also prove superconvergence results that guarantee smaller errors at special points in Ω. This phenomenon appears in our numerical examples in Section 5. In contrast, standard approaches solve for approximations to p and then numerically differentiate to compute $\mathbf{u} = -K \operatorname{grad} p$, thereby losing an order of accuracy in the velocity field [1].

To see the linear algebraic equations implied by (2.2), suppose $\mathbf{u}_h$ and p_h have the expansions

$$\mathbf{u}_h(x,y) = \left(\sum_{i=0}^{m} \sum_{j=1}^{n} U^x_{i,j} \phi^x_{i,j}(x,y), \sum_{i=1}^{m} \sum_{j=0}^{n} U^y_{i,j} \phi^y_{i,j}(x,y) \right),$$

$$p_h(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} \psi_{i,j}(x,y).$$

Here, $\phi^x_{i,j}$, $\phi^y_{i,j}$, and $\psi_{i,j}$ signify elements in the standard nodal bases for Q^x_h, Q^y_h, and V_h. Define the column vectors $U \in \mathbb{R}^{2mn+m+n}$, $P \in \mathbb{R}^{mn}$ containing the nodal unknowns as follows:

$$\begin{aligned} U^T = \; & (U^x_{0,1}, U^x_{1,1}, \ldots, U^x_{m,1}, \ldots, U^x_{0,n}, U^x_{1,n}, \ldots, U^x_{m,n}, \\ & U^y_{1,0}, U^y_{1,1}, \ldots, U^y_{1,n}, \ldots, U^y_{m,0}, U^y_{m,1}, \ldots, U^y_{m,n}), \\ P^T = \; & (P_{1,1}, P_{2,1}, \ldots, P_{m,1}, \ldots P_{1,n}, P_{2,n}, \ldots, P_{m,n}). \end{aligned} \tag{2.3}$$

Figure 1 shows how to associate these coefficients with nodes on a spatial grid Δ_h with $m = 4$, $n = 3$.

With these bases, the problem (2.2) has a matrix representation of the form

$$\begin{pmatrix} A & N \\ N^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} 0 \\ -F \end{pmatrix}. \tag{2.4}$$

Here A is a symmetric, positive definite matrix having the block structure

$$A = \begin{pmatrix} A^x & 0 \\ 0 & A^y \end{pmatrix},$$

in which $A^x \in \mathbb{R}^{(m+1)n\times(m+1)n}$ and $A^y \in \mathbb{R}^{m(n+1)\times m(n+1)}$ have entries of the form

$$\int_\Omega \frac{\phi^x_{i,j}\phi^x_{k,\ell}}{K}dxdy, \quad \int_\Omega \frac{\phi^y_{i,j}\phi^y_{k,\ell}}{K}dxdy,$$

respectively. Note that these entries contain information about the spatially varying coefficient K. The matrix N has the block structure

$$N = \begin{pmatrix} N^x \\ N^y \end{pmatrix},$$

where $N^x \in \mathbb{R}^{(m+1)n\times mn}$ and $N^y \in \mathbb{R}^{m(n+1)\times mn}$ have entries given, respectively, by

$$\int_\Omega \psi_{i,j}\frac{\partial\phi^x_{k,\ell}}{\partial x}dx\,dy, \quad \int_\Omega \psi_{i,j}\frac{\partial\phi^y_{k,\ell}}{\partial y}dx\,dy.$$

By calculating these integrals, one readily confirms that N^x and N^y reduce to the usual difference approximations to $\partial/\partial x$ and $\partial/\partial y$. The vector $F \in \mathbb{R}^{mn}$ has entries given by the integrals $\int_\Omega f\psi_{i,j}dx$. The Appendix to this paper gives more detail on the construction of A and N.

The matrix multiplying the nodal unknowns for $(\mathbf{u}_h, p_h)$ in (2.4) is not positive definite, but if we rewrite the system (2.4) as

$$AU = -NP,$$

$$-N^TA^{-1}NP = -F,$$

then A and its Schur complement $-N^TA^{-1}N$ are positive definite. In this sense, the system (2.4) is equivalent to two coupled, positive definite problems.

3 An h-Independent Iterative Method

Our first iterative scheme for solving the discrete system (2.4) is as follows:

Algorithm 1. *Beginning with initial guess* $\left(U^{(0)}, P^{(0)}\right)^T$ *for* (U, P), *the* k*-th iterate* $\left(U^{(k)}, P^{(k)}\right)^T$ *is the solution of*

$$\begin{pmatrix} \omega I & N \\ N^T & 0 \end{pmatrix}\begin{pmatrix} U^{(k)} \\ P^{(k)} \end{pmatrix} = \begin{pmatrix} 0 \\ -F \end{pmatrix} + \begin{pmatrix} \omega I - A & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} U^{(k-1)} \\ P^{(k-1)} \end{pmatrix}, \tag{3.1}$$

where $I \in \mathbb{R}^{(2mn+m+n)\times(2mn+m+n)}$ *is the identity matrix and* ω *is a parameter chosen to satisfy* $\omega \geq \rho(A)$.

Here, $\rho(A)$ denotes the spectral radius of the matrix A. Later in this section we discuss a practical way to pick ω that does not require detailed knowledge of the spectrum of A.

Computationally, Algorithm 1 has the following compact form: Given an initial guess $\left(U^{(0)}, P^{(0)}\right)^T$, compute $\left(U^{(k)}, P^{(k)}\right)^T$ by executing three steps:

$$(i) \quad G^{(k-1)} \leftarrow F + \omega^{-1} N^T (\omega I - A) U^{(k-1)}, \tag{3.2}$$

$$(ii) \quad \text{Solve } \omega^{-1} N^T N P^{(k)} = G^{(k-1)}, \tag{3.3}$$

$$(iii) \quad \omega U^{(k)} \leftarrow (\omega I - A) U^{(k-1)} - N P^{(k)}. \tag{3.4}$$

In each iteration, the main computational work is to solve for $P^{(k)} = \omega (N^T N)^{-1} G^{(k-1)}$. An easy calculation shows that the matrix $\omega^{-1}(N^T N)$ is positive definite, being proportional to the standard five-point, finite-difference Laplace operator applied to $P^{(k)}$. Therefore we expect the numerical solution for $P^{(k)}$ using stationary iterative methods to be plagued by poor conditioning when the grid mesh h is small. This observation leads us to use a multigrid scheme to get approximations to $P^{(k)}$. Such a device will preserve the h-independence of the overall scheme's convergence rate. We discuss this facet of the algorithm in more detail in the next section. For now let us analyze the convergence properties of the overall iterative scheme, assuming a "black-box" solver for $P^{(k)}$.

We begin by writing Equation (3.1) as a stationary iterative scheme:

$$\begin{pmatrix} U^{(k)} \\ P^{(k)} \end{pmatrix} = L + M \begin{pmatrix} U^{(k-1)} \\ P^{(k-1)} \end{pmatrix}, \tag{3.5}$$

where

$$L = \begin{pmatrix} \omega I & N \\ N^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ F \end{pmatrix},$$

$$M = \begin{pmatrix} \omega I & N \\ N^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \omega I - A & 0 \\ 0 & 0 \end{pmatrix}.$$

The convergence of Algorithm 1 depends on the spectral radius of the matrix M, for which the following proposition gives a bound.

Proposition 1. *Let*

$$0 < \lambda_{\min} \leq \cdots \leq \lambda_{\max} \tag{3.6}$$

be the eigenvalues of the matrix A, and let $\omega \geq \lambda_{\max}$. *Then the spectral radius of M obeys the estimate*

$$\rho(M) \leq 1 - \frac{\lambda_{\min}}{\omega}. \tag{3.7}$$

Proof: Let $\lambda \neq 0$ be an eigenvalue of M with eigenvector $(U_\lambda, P_\lambda)^T$. Thus

$$M \begin{pmatrix} U_\lambda \\ P_\lambda \end{pmatrix} \equiv \begin{pmatrix} \omega I & N \\ N^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \omega I - A & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U_\lambda \\ P_\lambda \end{pmatrix} = \lambda \begin{pmatrix} U_\lambda \\ P_\lambda \end{pmatrix}, \tag{3.8}$$

so

$$(\omega I - A)U_\lambda = \lambda(\omega U_\lambda + N P_\lambda), \tag{3.9a}$$

$$0 = \lambda N^T U_\lambda. \tag{3.9b}$$

Since $(U_\lambda, P_\lambda)^T \neq 0$, Equation (3.9a) shows that $U_\lambda \neq 0$; however, U_λ may be complex. Let U_λ^H denote its Hermitian conjugate. If we multiply (3.9a) by U_λ^H, observe that N is a real matrix, and apply (3.9b), we obtain

$$\begin{aligned} U_\lambda^H(\omega I - A)U_\lambda = &\ \lambda\omega U_\lambda^H U_\lambda + \lambda(N^T U_\lambda)^H P_\lambda \\ = &\ \lambda\omega U_\lambda^H U_\lambda. \end{aligned}$$

From this equation we conclude that

$$0 < |\lambda| = \left| \frac{U_\lambda^H(I - \omega^{-1}A)U_\lambda}{U_\lambda^H U_\lambda} \right| \leq \rho(I - \omega^{-1}A),$$

which implies

$$\rho(M) \leq \rho(I - \omega^{-1}A). \tag{3.10}$$

Also, by (3.6) and the fact that $\omega \geq \lambda_{\max}$, we have

$$\rho(I - \omega^{-1}A) \leq 1 - \frac{\lambda_{\min}}{\omega}.$$

These last two inequalities imply the desired bound (3.7). □

If we choose $\omega = \lambda_{\max} = \rho(A)$, then the estimate (3.7) for the spectral radius of the iteration matrix M becomes

$$\rho(M) \leq 1 - \frac{\lambda_{\min}}{\lambda_{\max}}.$$

To estimate $\lambda_{\min}/\lambda_{\max}$, the following proposition is helpful.

Proposition 2. *For the matrix A appearing in Equation (2.4), there exist constants* k_0 *and* k_1, *independent of h, such that*

$$k_0 h^2 U^T U \leq U^T A U \leq k_1 h^2 U^T U. \tag{3.11}$$

Proof: The representation of $\mathbf{u}_h$ given in Equation (2.3) leads to the identity

$$U^T A U = \int_\Omega \frac{1}{K} |\mathbf{u}_h|^2 dxdy = \sum_{i=1}^{m} \sum_{j=1}^{n} \int_{\Omega_{i,j}} \frac{1}{K} |\mathbf{u}_h|^2 dxdy,$$

where $\Omega_{i,j} = (x_{i-1}, x_i) \times (y_{j-1}, y_j)$. Since K is bounded and integrable on $\Omega_{i,j}$, the mean value theorem for integrals ([9], pp. 184-185) guarantees the existence of a number $K_{i,j}$, satisfying $\inf_{\Omega_{i,j}} K \leq K_{i,j} \leq \sup_{\Omega_{i,j}} K$, such that

$$\int_{\Omega_{i,j}} \frac{1}{K} |\mathbf{u}_h|^2 dxdy = \frac{1}{K_{i,j}} \int_{\Omega_{i,j}} |\mathbf{u}_h|^2 dxdy.$$

(If K^{-1} is continuous on $\bar{\Omega}_{i,j}$, then K^{-1} actually assumes the value $K_{i,j}^{-1}$ somewhere on $\Omega_{i,j}$.) Calculating the last integral using our basis for $\mathbf{Q}_h$, we get

$$\begin{aligned} U^T A U \; = & \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{a_{ij}}{6K_{ij}} \left[\begin{pmatrix} U^x_{i-1,j} \\ U^x_{i,j} \end{pmatrix}^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} U^x_{i-1,j} \\ U^x_{i,j} \end{pmatrix} \right. \\ & \left. + \begin{pmatrix} U^y_{i,j-1} \\ U^y_{i,j} \end{pmatrix}^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} U^y_{i,j-1} \\ U^y_{i,j} \end{pmatrix} \right], \end{aligned}$$

where a_{ij} signifies the area of $\Omega_{i,j}$. To simplify notation, we notice that the 2×2 matrix appearing in each term of this sum is positive definite. This observation allows us to define a new norm on $\mathbb{R}^2$ as follows:

$$\left\| \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} \right\|_A^2 = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}.$$

If $\|\cdot\|_2$ denotes the usual Euclidean norm on $\mathbb{R}^2$, then it is easy to check that $\|\cdot\|_2^2 \leq \|\cdot\|_A^2 \leq 3\|\cdot\|_2^2$. In terms of the new norm,

$$U^T A U = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{a_{ij}}{6K_{ij}} \left[\left\| \begin{pmatrix} U^x_{i-1,j} \\ U^x_{i,j} \end{pmatrix} \right\|_A^2 + \left\| \begin{pmatrix} U^y_{i,j-1} \\ U^y_{i,j} \end{pmatrix} \right\|_A^2 \right].$$

The quantity $U^T U$ is easier to calculate:

$$U^T U = \sum_{i=0}^{m} \sum_{j=1}^{n} |U^x_{ij}|^2 + \sum_{i=1}^{m} \sum_{j=0}^{n} |U^y_{ij}|^2. \tag{3.12}$$

Now we use the bounds on K and the quasiuniformity of Δ_h to observe that

$$\begin{aligned} U^T A U &\geq \frac{\alpha^2 h^2}{6K_{\sup}} \sum_{i=1}^{m}\sum_{j=1}^{n} \left[\left\| \begin{pmatrix} U^x_{i-1,j} \\ U^x_{i,j} \end{pmatrix} \right\|_A^2 + \left\| \begin{pmatrix} U^y_{i,j-1} \\ U^y_{i,j} \end{pmatrix} \right\|_A^2 \right] \\ &\geq \frac{\alpha^2 h^2}{6K_{\sup}} \sum_{i=1}^{m}\sum_{j=1}^{n} \left[\left\| \begin{pmatrix} U^x_{i-1,j} \\ U^x_{i,j} \end{pmatrix} \right\|_2^2 + \left\| \begin{pmatrix} U^y_{i,j-1} \\ U^y_{i,j} \end{pmatrix} \right\|_2^2 \right] \\ &\geq \frac{\alpha^2 h^2}{6K_{\sup}} \left[\sum_{i=0}^{m}\sum_{j=1}^{n} \left\| \begin{pmatrix} 0 \\ U^x_{i,j} \end{pmatrix} \right\|_2^2 + \sum_{i=1}^{m}\sum_{j=0}^{n} \left\| \begin{pmatrix} 0 \\ U^y_{i,j} \end{pmatrix} \right\|_2^2 \right] \\ &= \frac{\alpha^2 h^2}{6K_{\sup}} U^T U. \end{aligned}$$

This observation establishes the first inequality in (3.11), since we can take $k_0 = \alpha^2/6K_{\sup}$. To prove the second inequality in (3.11), we rewrite Equation (3.12) as follows:

$$\begin{aligned} U^T U = &\frac{1}{2} \sum_{i=0}^{m}\sum_{j=0}^{n} \left[\left\| \begin{pmatrix} 0 \\ U^x_{i,j} \end{pmatrix} \right\|_2^2 + \left\| \begin{pmatrix} 0 \\ U^y_{i,j} \end{pmatrix} \right\|_2^2 \right] \\ &+ \frac{1}{2} \sum_{i=1}^{m+1}\sum_{j=1}^{n+1} \left[\left\| \begin{pmatrix} U^x_{i-1,j} \\ 0 \end{pmatrix} \right\|_2^2 + \left\| \begin{pmatrix} U^y_{i,j-1} \\ 0 \end{pmatrix} \right\|_2^2 \right], \end{aligned}$$

where we agree that $U^x_{ij} = 0$ if either $j = 0$ or $j = n+1$, and $U^y_{ij} = 0$ if either $i = 0$ or $i = m+1$. Hence,

$$\begin{aligned} U^T U &\geq \frac{1}{2} \sum_{i=1}^{m}\sum_{j=1}^{n} \left[\left\| \begin{pmatrix} U^x_{i-1,j} \\ U^x_{i,j} \end{pmatrix} \right\|_2^2 + \left\| \begin{pmatrix} U^y_{i,j-1} \\ U^y_{i,j} \end{pmatrix} \right\|_2^2 \right] \\ &\geq \frac{1}{6} \sum_{i=1}^{m}\sum_{j=1}^{n} \left[\left\| \begin{pmatrix} U^x_{i-1,j} \\ U^x_{i,j} \end{pmatrix} \right\|_A^2 + \left\| \begin{pmatrix} U^y_{i,j-1} \\ U^y_{i,j} \end{pmatrix} \right\|_A^2 \right] \\ &\geq \frac{K_{\inf}}{h^2} U^T A U. \end{aligned}$$

We conclude that $U^T A U \leq k_1 h^2 U^T U$, where $k_1 = 1/K_{\inf}$. □

If we apply Proposition 2 to the case when U is an eigenvector of A associated with the eigenvalue $\lambda_{\min}$ or $\lambda_{\max}$, respectively, we find that $\lambda_{\min} \geq \alpha^2 h^2/6K_{\sup}$ and $\lambda_{\max} \leq h^2/K_{\inf}$. Therefore, provided we choose $\omega \geq \lambda_{\max}$ in Algorithm 1, the spectral radius of our iteration matrix M obeys the bound

$$\rho(M) \leq 1 - \frac{\alpha^2 K_{\inf}}{6K_{\sup}}. \tag{3.13}$$

Notice that the right side of this inequality is a constant independent of h. This is the sense in which the convergence rate of Algorithm 1 is independent of h.

Two remarks about the practical implications of the estimate (3.13) are in order. First, bound on $\rho(M)$ depends strongly on the nature of the coefficient $K(x,y)$. In particular, if $K_{\inf}/K_{\sup}$ is very small, reflecting a high degree of heterogeneity in the physical problem, then we can expect the actual convergence of the algorithm to be slow, albeit independent of grid mesh. Several examples in Section 5 confirm this expectation. Second, even though choosing $\omega = \lambda_{\max}$ minimizes $\rho(M)$ and hence optimizes the convergence rate, this choice is impractical owing to the expense of calculating $\lambda_{\max}$. In practice, we typically pick $\omega = \|A\|_\infty \geq \lambda_{\max}$. This choice is easily computable as the maximum row sum of A, and it preserves h-independence of convergence rate even though it may be theoretically nonoptimal.

4 Application of a Multigrid Solver

As we have mentioned, the computation of the pressure iterate $P^{(k)}$ in step (ii) of Algorithm 1 is inefficient if we use direct schemes or classical stationary iterative methods on fine grids. However, the fact that $\omega^{-1}N^TN$ is essentially the finite-difference Laplacian operator motivates us to reduce the computational work for each iteration by calculating an approximation to the k-th pressure iterate by using several cycles of multigrid method on the system (3.3). We refer the reader to [3] for a discussion of the multigrid approach and for a Fortran code applicable in the context of our problem. The modified scheme is as follows:

Algorithm 2. *Begin with an initial guess* $\left(U^{(0)}, P^{(0)}\right)^T$, *and suppose that we have computed* $\left(U^{(k-1)}, P^{(k-1)}\right)^T$. *Compute a new approximation* $\left(U^{(k)}, P^{(k)}\right)^T$ *using the following steps:*

1. *Compute the residual,*

$$G^{(k-1)} = F + N^T(I - \omega^{-1}A)U^{(k-1)}. \tag{4.1}$$

2. *Let* $\hat{P}^{(k)}$ *denote the exact solution of the problem*

$$\omega^{-1}N^TN\hat{P}^{(k)} = G^{(k-1)}. \tag{4.2}$$

Calculate an approximation $P^{(k)}$ of $\hat{P}^{(k)}$ by applying r cycles of the multigrid algorithm [3] to the equation (4.2), using $P^{(k-1)}$ as initial guess. (We discuss the choice of r below.)

3. *Compute $U^{(k)}$ as in Algorithm 1:*

$$\omega U^{(k)} = (\omega I - A)U^{(k-1)} - NP^{(k)}. \tag{4.3}$$

Multigrid methods for solving elliptic problems have an advantage that is quite relevant to the conditioning problems associated with fine grids: Each cycle has a convergence rate that is independent of h ([4], Chapter 4). Therefore, we need only show that we can choose a *fixed* number r of multigrid cycles such that each iteration of Algorithm 2 reduces the error norm by an appropriate factor close to $\rho(M)$. We do this in Proposition 3. Since the factor is independent of h, Algorithm 2 has convergence rate independent of h.

We begin by defining norms on the "pressure" and "velocity" spaces that will make the proof easier. Any $p_h \in V_h$ has a representation

$$p_h(x,y) = \sum_{i,j} P_{i,j}\psi_{i,j}(x,y).$$

Taking advantage of the fact that N^TN is positive definite, we compute a norm of the vector

$$P = (P_{1,1}, P_{2,1}, \ldots, P_{m,1}, \ldots, P_{1,n}, P_{2,n}, \ldots, P_{m,n})^T$$

by setting $||P||_h^2 = P^T(\omega^{-1}N^TN)P$. On the other hand, any $\mathbf{u}_h \in \mathbf{Q}_h$ has a representation

$$\mathbf{u}_h(x,y) = \left(\sum_{i,j} U^x_{i,j}\phi^x_{i,j}(x,y), \sum_{i,j} U^y_{i,j}\phi^y_{i,j}(x,y)\right).$$

We compute a norm of the vector

$$\begin{aligned} U = \big(&U^x_{0,1}, U^x_{1,1}, \ldots, U^x_{m,1}, \ldots, U^x_{0,n}, U^x_{1,n}, \ldots, U^x_{m,n}, \\ &U^y_{1,0}, U^y_{1,1}, \ldots, U^y_{1,n}, \ldots, U^y_{m,0}, U^y_{m,1}, \ldots, U^y_{m,n}\big)^T \end{aligned}$$

by setting $||U||_\omega^2 = \omega U^TU$.

The norm $\|\cdot\|_\omega$ is just a scalar multiple of the Euclidean distance function $\|\cdot\|_2$, and since ω is a constant related to $\rho(A)$, $\|\cdot\|_\omega$ is actually a discrete analog of the Euclidean

norm $\|\cdot\|_{L^2(\Omega)\times L^2(\Omega)}$ on the velocity space by Proposition 2. This norm is appropriate for measuring the convergence of velocity iterates $U^{(k)}$ to the true discrete approximation U. Also, since N^TN is just the positive definite matrix associated with the five-point difference approximation to the Laplace operator, the norm $\|\cdot\|_h$ is appropriate for measuring the rapidity with which the pressure iterates satisfy the discrete pressure equation (3.3) as the iterations progress. Ultimately, we want to relate our results to more familiar norms such as $\|\cdot\|_2$ and $\|\cdot\|_\infty$; for this step we shall rely on the equivalence of norms for finite-dimensional Euclidean spaces.

In the following proposition, we assume $\nu = \rho(I - \omega^{-1}A) < 1$. Thus ν is an upper bound on $\rho(M)$. Suppose the multigrid iteration used to approximate $\hat{P}^{(k)}$ in step 2 of Algorithm 1 has convergence rate $\mu \in (0,1)$. This implies that, after r multigrid cycles for $P^{(k)}$ using $P^{(k-1)}$ as initial guess,

$$\left\|\hat{P}^{(k)} - P^{(k)}\right\|_h \leq \mu^r \left\|\hat{P}^{(k)} - P^{(k-1)}\right\|_h . \tag{4.4}$$

Proposition 3. *For any $\nu' \in (\nu, 1)$, there exists a number r of multigrid cycles such that*

$$\left\|P - P^{(k)}\right\|_h + \left\|U - U^{(k)}\right\|_\omega \leq \nu' \left(\left\|P - P^{(k-1)}\right\|_h + \left\|U - U^{(k-1)}\right\|_\omega\right),$$

where (P,U) is the solution of the problem (2.4) and $\left(P^{(k)}, U^{(k)}\right)$ is the approximation to (P,U) produced by the k-th iteration of Algorithm 3.

Proof: Suppose we compute $\hat{U}^{(k)}$ according to (3.4) with the exact (nonmultigrid) pressure iterate $\hat{P}^{(k)}$. Thus,

$$\omega\hat{U}^{(k)} = (\omega I - A)U^{(k-1)} - N\hat{P}^{(k)}, \tag{4.5}$$

where $\hat{P}^{(k)}$ satisfies Equation (4.2). Then from Equations (2.4), (4.1), (4.2) and (4.5), we have

$$\omega\left(U - \hat{U}^{(k)}\right) + N\left(P - \hat{P}^{(k)}\right) = (\omega I - A)\left(U - U^{(k-1)}\right), \tag{4.6}$$

$$N^T\left(U - \hat{U}^{(k)}\right) = 0. \tag{4.7}$$

Multiplying Equation (4.6) by $\left(U - \hat{U}^{(k)}\right)^T$ and using the identity (4.7), we get

$$\begin{aligned}
\left\|U - \hat{U}^{(k)}\right\|_\omega^2 &= \left(U - \hat{U}^{(k)}\right)^T (\omega I - A)\left(U - U^{(k-1)}\right) \\
&\leq \left\|U - \hat{U}^{(k)}\right\|_\omega \left\|(I - \omega^{-1}A)\left(U - U^{(k-1)}\right)\right\|_\omega \\
&\leq \rho(I - \omega^{-1}A)\left\|U - \hat{U}^{(k)}\right\|_\omega \left\|U - U^{(k-1)}\right\|_\omega .
\end{aligned}$$

Therefore, the velcity iterates obey the estimate

$$\left\|U - \hat{U}^{(k)}\right\|_\omega \leq \nu \left\|U - U^{(k-1)}\right\|_\omega .$$

Similarly, multiplying Equation (4.6) by $\left[\omega^{-1} N \left(P - \hat{P}^{(k)}\right)\right]^T$, we get

$$\begin{aligned}
\left\|P - \hat{P}^{(k)}\right\|_h^2 = & \left(P - \hat{P}^{(k)}\right)^T N^T \omega^{-1} (\omega I - A) \left(U - U^{(k-1)}\right) \\
\leq & \left\|\omega^{-1} N \left(P - \hat{P}^{(k)}\right)\right\|_\omega \left\|(I - \omega^{-1} A) \left(U - U^{(k-1)}\right)\right\|_\omega \\
\leq & \left\|P - \hat{P}^{(k)}\right\|_h \rho(I - \omega^{-1} A) \left\|U - U^{(k-1)}\right\|_\omega .
\end{aligned}$$

Hence, the pressure iterates obey the bound

$$\left\|P - \hat{P}^{(k)}\right\|_h \leq \nu \left\|U - U^{(k-1)}\right\|_\omega .$$

Now we derive bounds on $\left\|P - P^{(k)}\right\|_h$ and $\left\|U - U^{(k)}\right\|_\omega$ in terms of their values at the previous iterative level. For $\left\|P - P^{(k)}\right\|_h$, we use the triangle equality and the multigrid estimate (4.4) to get

$$\begin{aligned}
\left\|P - P^{(k)}\right\|_h \leq & \left\|P - \hat{P}^{(k)}\right\|_h + \left\|\hat{P}^{(k)} - P^{(k)}\right\|_h \\
\leq & \left\|P - \hat{P}^{(k)}\right\|_h + \mu^r \left\|\hat{P}^{(k)} - P^{(k-1)}\right\|_h \qquad (4.8) \\
\leq & \left\|P - \hat{P}^{(k)}\right\|_h + \mu^r \left(\left\|P - \hat{P}^{(k)}\right\|_h + \left\|P - P^{(k-1)}\right\|_h\right) .
\end{aligned}$$

But the original iterative scheme (3.5) implies

$$\begin{pmatrix} U - \hat{U}^{(k)} \\ P - \hat{P}^{(k)} \end{pmatrix} = M \begin{pmatrix} U - U^{(k-1)} \\ P - P^{(k-1)} \end{pmatrix} .$$

So, in light of the inequality (3.1) bounding $\rho(M)$ by ν, we have

$$\left\|\hat{P} - P^{(k)}\right\|_h \leq \rho(M) \left\|P - P^{(k-1)}\right\|_h \leq \nu \left\|P - P^{(k-1)}\right\|_h .$$

This inequality allows us to simplify (4.8), getting

$$\left\|P - P^{(k)}\right\|_h \leq (\nu + \mu^r + \nu\mu^r) \left\|P - P^{(k-1)}\right\|_h . \qquad (4.9)$$

Turning to $\left\|U - U^{(k)}\right\|_\omega$, we use Equation (4.3), multiplied by ω^{-1}, to write

$$\left(U - U^{(k)}\right) = (I - \omega^{-1} A) \left(U - U^{(k-1)}\right) + \omega^{-1} N \left(P - P^{(k)}\right) .$$

This identity implies

$$\begin{aligned}\left\|U-U^{(k)}\right\|_\omega &\le \left\|(I-\omega^{-1}A)\left(U-U^{(k-1)}\right)\right\|_\omega + \left\|\omega^{-1}N\left(P-P^{(k)}\right)\right\|_\omega \\ &\le \nu\left\|U-U^{(k-1)}\right\|_\omega + \left\|P-P^{(k)}\right\|_h \\ &\le \nu\left\|U-U^{(k-1)}\right\|_\omega + (\nu+\mu^r+\nu\mu^r)\left\|P-P^{(k-1)}\right\|_h \\ &\le (\nu+\mu^r+\nu\mu^r)\left(\left\|P-P^{(k-1)}\right\|_h + \left\|U-U^{(k-1)}\right\|_\omega\right).\end{aligned} \tag{4.10}$$

Combining the inequalities (4.9) and (4.10), we get

$$\begin{aligned}&\left\|P-P^{(k)}\right\|_h + \left\|U-U^{(k)}\right\|_\omega \\ &\le (\nu+\mu^r+\nu\mu^r)\left(\left\|P-P^{(k)}\right\|_h + \left\|U-U^{(k-1)}\right\|_\omega\right).\end{aligned}$$

Since $\mu < 1$, $\mu^r + \nu\mu^r \to 0$ as $r \to \infty$. We can therefore choose r large enough so that $\nu + \mu^r + \nu\mu^r + \nu \le \nu' < 1$. In this way,

$$\left\|P-P^{(k)}\right\|_h + \left\|U-U^{(k)}\right\|_\omega \le \nu'\left(\left\|P-P^{(k-1)}\right\|_h + \left\|U-U^{(k-1)}\right\|_\omega\right). \quad \square$$

In view of the norm equivalence mentioned earlier, Proposition 3 leads us to expect that, if we choose ω as prescribed in Section 3, then the computed convergence rate

$$\bar{\mu} = \lim_{k\to\infty}\left[\frac{\|P-P^{(k)}\|_\infty + \|U-U^{(k)}\|_\infty}{\|P-P^{(0)}\|_\infty + \|U-U^{(0)}\|_\infty}\right]^{1/k} \tag{4.11}$$

should be a constant independent of h as $h \to 0$. In fact, for "generic" initial guesses, the contribution from the eigenvector associated with the largest-magnitude eigenvalue of M will eventually dominate the error. We therefore expect $\bar{\mu}$ to give good approximations to $\rho(M)$ in computational practice ([2], p. 129).

5 Numerical Examples of h-Independence

To test our results, we apply Algorithm 2 to several versions of the following boundary-value problem:

$$\begin{aligned}-\mathrm{div}\left[K(x,y)\mathrm{grad}\, p(x,y)\right] &= f(x,y), \qquad (x,y)\in\Omega, \\ p(x,y) &= 0, \qquad (x,y)\in\partial\Omega.\end{aligned} \tag{5.1}$$

We use the lowest-order mixed finite-element method on grids with $h = 2^{-\ell}$, where $\ell = 4,5,6,7,8$. Each iteration of the solution scheme includes $r = 2$ V-cycles of the multigrid

algorithm described in [3], where the coarsest grid in each cycle has mesh 2^{-1}, and the finest has mesh $2^{-\ell}$. We use the following realizations of the coefficient $K(x,y)$:

$$\begin{aligned} K_{\mathrm{I}}(x,y) &= 1; \\ K_{\mathrm{II}}(x,y) &= e^{-x-y}; \\ K_{\mathrm{III}}(x,y) &= \begin{cases} 1, & \text{if } x < y, \\ 0.1, & \text{if } x \geq y; \end{cases} \\ K_{\mathrm{IV}}(x,y) &= K_{\mathrm{II}}(x,y) \cdot K_{\mathrm{III}}(x,y); \\ K_{\mathrm{V}}(x,y) &= \begin{cases} 1, & \text{if } x < y, \\ 0.01, & \text{if } x \geq y. \end{cases} \end{aligned}$$

To confirm the convergence properties of the mixed finite-element method as $h \to 0$, we examine the exact and numerical solutions to (5.1) using $K = K_{\mathrm{II}}$ and taking $f(x,y)$ to be the function that results when the solution is $p(x,y) = x(1-x)\sin(\pi y) + y(1-y)\sin(\pi x)$. We compute the nodal error indicators $\|U_{\text{exact}} - U\|_\infty$ and $\|P_{\text{exact}} - P\|_\infty$, where U_{exact} and P_{exact} stand for the vectors of nodal values of the exact solutions $\mathbf{u}$ and p, and U and P are vectors containing nodal values of the finite-element approximations on a uniform grid of mesh h. Figure 2 shows plots of $\log \|U_{\text{exact}} - U\|_\infty$ and $\log \|P_{\text{exact}} - P\|_\infty$ versus $\log h$ having least-squares slopes of 1.899 and 2.000, respectively. These results suggest that the nodal values of U and P are accurate to $O(h^2)$, corroborating the equal-order accuracy available in the Raviart-Thomas subspaces and indicating superconvergent nodal values in accordance with the work of Ewing et al. [6].

To check the convergence properties of the iterative scheme, we examine the behavior of the ratio $\bar{\mu}$, defined in Equation (4.11), for each of the choices of K. Our results, shown in Figure 3, support the expectation that, as $h \to 0$, the convergence rate of the scheme tends to a constant independent of h. Notice however that, as K exhibits more spatial variation, the convergence of the algorithm becomes slower. Any effects of variability in K on the conditioning of the discrete equations still influence this first algorithm; the only effects of poor conditioning that we have eliminated so far are those associated with grid refinement.

6 Modified Schemes for Heterogeneous Media

To mitigate the difficulties associated with spatial variability, we modify the first iterative scheme (3.1) to get a class of new schemes having the following form:

Algorithm 3. *Beginning with initial guess* $\left(U^{(0)}, P^{(0)}\right)^T$, *the k-th iterate* $\left(U^{(k)}, P^{(k)}\right)^T$ *is the solution of*

$$\begin{pmatrix} D & N \\ N^T & 0 \end{pmatrix} \begin{pmatrix} U^{(k)} \\ P^{(k)} \end{pmatrix} = \begin{pmatrix} 0 \\ -F \end{pmatrix} + \begin{pmatrix} D-A & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U^{(k-1)} \\ P^{(k-1)} \end{pmatrix}. \tag{6.1}$$

Here, the "preconditioning" matrix $D \in \mathbb{R}^{(2mn+m+n)\times(2mn+m+n)}$ *is a diagonal matrix whose choice we discuss below.*

When we construct D properly, the iteration matrix

$$M = \begin{pmatrix} D & N \\ N^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} D-A & 0 \\ 0 & 0 \end{pmatrix} \tag{6.2}$$

has spectral radius that is independent of both h and the structure of K. The price we pay for this benefit is apparent in the computational form of the new algorithm:

$$(i) \quad G^{(k-1)} \leftarrow F + N^T D^{-1}(D-A)U^{(k-1)}, \tag{6.3}$$

$$(ii) \quad \text{Solve } N^T D^{-1} N P^{(k)} = G^{(k-1)}, \tag{6.4}$$

$$(iii) \quad U^{(k)} \leftarrow D^{-1}(D-A)U^{(k-1)} - D^{-1}NP^{(k)}. \tag{6.5}$$

In contrast to Equation (3.3), solving for $P^{(k)}$ in the new scheme calls for the inversion of $N^T D^{-1} N$ instead of $N^T N$. Therefore, we must modify the multigrid segment of the algorithm to accommodate variable coefficients. As we discuss, this modification is fairly easy to make. This section establishes criteria for the construction of D, gives two examples that satisfy these criteria, comments on the the multigrid solver used, and presents computational results.

As with the original scheme presented in Section 3, the key to the convergence of the new scheme is the spectral radius of the iteration matrix M defined in Equation (6.2). The following proposition gives sufficient conditions under which $\rho(M) < 1$.

Proposition 4. *Suppose D is a diagonal matrix with positive entries on the diagonal, and suppose there exist constants $b_1, b_2 \in (0,1)$ such that*

$$b_1 \leq \frac{U^H A U}{U^H D U} \leq 2 - b_2$$

for all vectors $U \in \mathbb{C}^{(m+1)n+m(n+1)}$. Then the iteration matrix M defined in Equation (6.2) satisfies

$$0 < \rho(M) \leq \max\{1 - b_1, 1 - b_2\} < 1. \tag{6.6}$$

Proof: Let $\lambda \neq 0$ be an eigenvalue of M with associated eigenvector $(U_\lambda, P_\lambda)^T$, as in Proposition 1. Then steps similar to those yielding Equations (3.9) show that

$$\begin{aligned} (D - A)U_\lambda &= \lambda(DU_\lambda + NP_\lambda), \\ 0 &= \lambda N^T U_\lambda. \end{aligned}$$

Thus $U_\lambda^H (D - A) U_\lambda = \lambda U_\lambda^H D U_\lambda$, which is nonzero since D is positive definite. Therefore,

$$|\lambda| = \left| 1 - \frac{U_\lambda^H A U_\lambda}{U_\lambda^H D U_\lambda} \right|.$$

Hence, using the hypothesized bounds on $U_\lambda^H A U_\lambda / U_\lambda^H D U_\lambda$, we have the desired inequalities (6.6). □

To use this proposition, we need estimates on $U^H A U$. Given the structure of A as shown in the Appendix, one can calculate a useful expression for $U^H A U$, assuming $U \in \mathbb{C}^{(m+1)n+m(n+1)}$ has the form $(U^x, U^y)^T$ indicated in Equation (2.3). In particular,

$$U^H A U = \frac{1}{3} S(U) + \frac{1}{6} R(U),$$

where, in the notation of the Appendix,

$$S(U) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(T_{i,j}^{\mathrm{I}} |U_{i-1,j}^x|^2 + T_{i,j}^{\mathrm{III}} |U_{i,j}^x|^2 + T_{i,j}^{\mathrm{IV}} |U_{i,j-1}^y|^2 + T_{i,j}^{\mathrm{VI}} |U_{i,j}^y|^2 \right),$$

$$R(U) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[T_{i,j}^{\mathrm{II}} \left(\bar{U}_{i,j}^x U_{i-1,j}^x + \bar{U}_{i-1,j}^x U_{i,j}^x \right) + T_{i,j}^{\mathrm{V}} \left(\bar{U}_{i,j}^y U_{i,j-1}^y + \bar{U}_{i,j-1}^y U_{i,j}^y \right) \right].$$

Here, $\bar{z}$ denotes the complex conjugate of z. The coefficients $T_{i,j}^{\mathrm{I}}, \ldots, T_{i,j}^{\mathrm{VI}}$ appearing in these expressions are values depending on $K(x,y)$ and arising from applications of the mean value theorem for integrals over each cell $\bar{\Omega}_{i,j}$ in the finite-element grid Δ_h. By using the inequalities

$$\begin{aligned} |w|^2 + |z|^2 &\geq w\bar{z} + \bar{w}z = |w|^2 + |z|^2 - |w - z|^2 \\ &\geq |w|^2 + |z|^2 - (|w| - |z|)^2 = -2|w||z|, \end{aligned}$$

we can estimate $R(U)$ as follows:

$$-2\sum_{i=1}^{m}\sum_{j=1}^{n}\left(T^{\mathrm{II}}_{i,j}|U^x_{i,j}||U^x_{i-1,j}| + T^{\mathrm{V}}_{i,j}|U^y_{i,j}||U^y_{i,j-1}|\right) \le R(U)$$

$$\le \sum_{i=1}^{m}\sum_{j=1}^{n}\left(T^{\mathrm{II}}_{i,j}|U^x_{i-1,j}|^2 + T^{\mathrm{II}}_{i,j}|U^x_{i,j}|^2 + T^{\mathrm{V}}_{i,j}|U^y_{i,j-1}|^2 + T^{\mathrm{V}}_{i,j}|U^y_{i,j}|^2\right). \tag{6.7}$$

In general, the estimates $0 < K_{\mathrm{inf}} \le K \le K_{\mathrm{sup}}$ may be too coarse to provide enough control on the coefficients $T^{\mathrm{I}}_{i,j},\ldots,T^{\mathrm{VI}}_{i,j}$ for constructing a reasonable preconditioner D. Strictly speaking, the necessary level of control will be available only if we have information about the *local* variation of K on each cell $\bar{\Omega}_{i,j}$.

In practice, however, we rarely have such fine-scale knowledge of K, and even if we did we would not try use it in calculating the Galerkin integrals $\int_\Omega K^{-1}\mathbf{u}\cdot\mathbf{v}dxdy$ exactly. Instead, most practical codes use approximate quadrature schemes that effectively treat K^{-1} as piecewise polynomial. In fact, it is quite common to treat K^{-1} as piecewise constant. In such applications, we can use the second inequality in (6.7), together with the identities $T^{\mathrm{II}}_{i,j} = T^{\mathrm{V}}_{i,j} = T_{i,j}$, to show that

$$U^H AU = \frac{1}{3}S(U) + \frac{1}{6}R(U) \le \frac{1}{2}S(U).$$

Similarly, the first inequality in (6.7), together with the identities $T^{\mathrm{I}}_{i,j} = T^{\mathrm{III}}_{i,j} = T^{\mathrm{IV}}_{i,j} = T^{\mathrm{VI}}_{i,j} = T_{i,j}$, shows that

$$\begin{aligned} U^H AU &= \frac{1}{6}S(U) + \frac{1}{6}[S(U) + R(U)] \\ &\ge \frac{1}{6}S(U) + \frac{1}{6}\sum_{i=1}^{m}\sum_{j=1}^{n}T_{i,j}\left[\left(|U^x_{i-1,j}| - |U^x_{i,j}|\right)^2 + \left(|U^y_{i,j-1}| - |U^y_{i,j}|\right)^2\right] \\ &\ge \frac{1}{6}S(U). \end{aligned}$$

In summary, $\frac{1}{6}S(U) \le U^H AU \le \frac{1}{2}S(U)$ whenever K is piecewise constant on the grid Δ_h.

Now consider the choice $D = \frac{2}{3}\mathrm{lump}(A)$, where

$$[\mathrm{lump}(A)]_{i,j} = \begin{cases} 0, & \text{if } i \ne j, \\ \sum_j A_{i,j}, & \text{if } i = j. \end{cases}$$

This is the matrix that results when we add entries along each row of A and assign the sum to the diagonal entry in that row. This choice of D is a simple instance of a

preconditioner developed in [7] for other iterative schemes. It is a straightforward matter to show that, when K is piecewise constant, $U^H \text{lump}(A) U = \frac{1}{2} S(U)$, so $U^H D U = \frac{1}{3} S(U)$. As a consequence,

$$b_1 = \frac{1}{2} \leq \frac{U^H A U}{U^H D U} \leq \frac{3}{2} = 2 - b_2.$$

Therefore, by Proposition 4, $\rho(M) \leq \frac{1}{2}$, and the iterative scheme converges with a rate independent of h and K. According to our remarks at the end of Section 4, we expect the ratio of error norms between sucessive iterates to approach $\frac{1}{2}$ as the iteration counter $k \to \infty$.

As an even simpler example, consider the choice $D = \text{diag}(A)$, where

$$[\text{diag}(A)]_{i,j} = \begin{cases} 0, & \text{if } i \neq j, \\ A_{i,i}, & \text{if } i = j, \end{cases}$$

is the matrix A stripped of its off-diagonal entries. This choice has the attractive feature that it is trivial to compute from A. With D defined in this way, we once again find that $U^H D U = \frac{1}{3} S(U)$ when K is piecewise constant on Δ_h. Therefore, $\rho(M) \leq \frac{1}{2}$, and this iterative scheme also converges with a rate independent of h and K.

Either choice of D requires us to solve a matrix equation of the form $N^T D^{-1} N P^{(k)} = G^{(k-1)}$ at each iteration. To do this, we use two cycles of a multigrid scheme in which the Jacobi iteration is the smoother, the coarse-to-fine interpolation is bilinear, and the fine-to-coarse restriction is accomplished using half-injection ([4], p. 65). This scheme preserves the h-independence of the overall algorithm's convergence rate and appears to handle variable the variable coefficient K effectively. Alternative multigrid implementations are certainly possible here.

To test the convergence rate of Algorithm 3, we apply it to the boundary-value problems described in Section 5, using the preconditioner $D = \frac{2}{3} \text{lump}(A)$. Table 1 shows values of the convergence rate $\bar{\mu}$ computed for each choice of coefficient K, for each of five different values of the grid mesh h. All of the tabulated values are very close to the spectral radius estimate $\rho(M) \leq \frac{1}{2}$. We conclude that this scheme converges at a rate independent of both grid mesh h and the heterogeneity reflected in the mobility coefficient K.

7 Acknowledgments

The authors wish to acknowledge the following sources of support: NSF grant RII-8610680, ONR grant 0014-88-K-0370, and the Wyoming Water Research Center.

8 References

[1]. Allen, M.B., Ewing, R.E., and Koebbe, J.V., "Mixed finite-element methods for computing groundwater velocities," *Numer. Meth. P.D.E. 3* (1985), 195-207.

[2]. Birkhoff, G., and Lynch, R.E., *Numerical Solution of Elliptic Problems*, Philadelphia: SIAM, 1984.

[3]. Brandt, A., "Multi-level adaptive solutions to boundary-value problems," *Math. Comp. 31:138* (1977), 333-390.

[4]. Briggs, W.L., *A Multigrid Tutorial*, Philadelphia: SIAM, 1987.

[5]. Douglas, J., Ewing, R.E., and Wheeler, M.F., "The approximation of the pressure by a mixed method in the simulation of miscible displacement," *R.A.I.R.O. Analyse Numerique 17* (1983), 17-33.

[6]. Ewing, R.E., Lazarov, R.D., and Wang, J., "Superconvergence of the velocities along the Gaussian lines in the mixed finite element method," to appear in *SIAM J. Numer. Anal.*

[7]. Ewing, R.E., Lazarov, R.D., Lu, P., and Vassilevski, P.S., "Preconditioning indefinite systems arising from mixed finite-element discretizations of second-order elliptic systems," in *Proceedings, Conference on Preconditioned Conjugate Gradient Methods*, Nijmegen, The Netherlands, June 15-17, 1989 (to appear).

[8]. Ewing, R.E. and Wheeler, M.F., "Computational aspects of mixed finite element methods," in *Numerical Methods for Scientific Computing,* ed. by R.S. Stepelman, Amsterdam: North-Holland, 1983, 163-172.

[9]. Munroe, M.E., *Introduction to Measure and Integration*, Cambridge, MA: Addison-Wesley, 1953.

[10]. Raviart, P.A. and Thomas, J.M., "A mixed finite element method for 2nd order elliptic problems," in *Mathematical Aspects of Finite Element Methods* (Lecture Notes in

Mathematics vol. 606), ed. by I. Galligani and E. Magenes, Berlin: Springer-Verlag, 1977, 292-315.

TABLE 1. CONVERGENCE RATES FOR VARIOUS COEFFICIENTS AND GRIDS.

	GRID MESH h				
COEFFICIENT	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
K_{I}	0.4933	0.4988	0.4993	0.4995	0.4999
K_{II}	0.4966	0.4995	0.4988	0.4997	0.4999
K_{III}	0.4948	0.4982	0.4991	0.4998	0.4999
K_{IV}	0.4947	0.4980	0.4992	0.4998	0.4999
K_{V}	0.4939	0.4978	0.4989	0.4999	0.5000

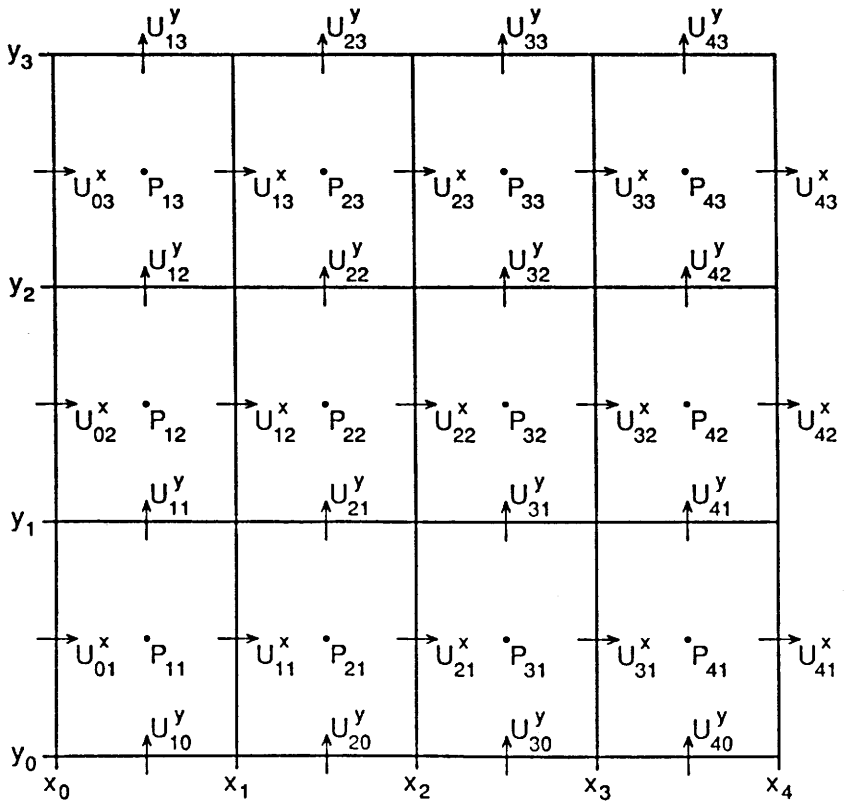


Figure 1. Sample 4×3 rectangular grid on $\Omega = (0,1) \times (0,1)$, showing locations of the nodal unknowns in the velocity and pressure trial functions.

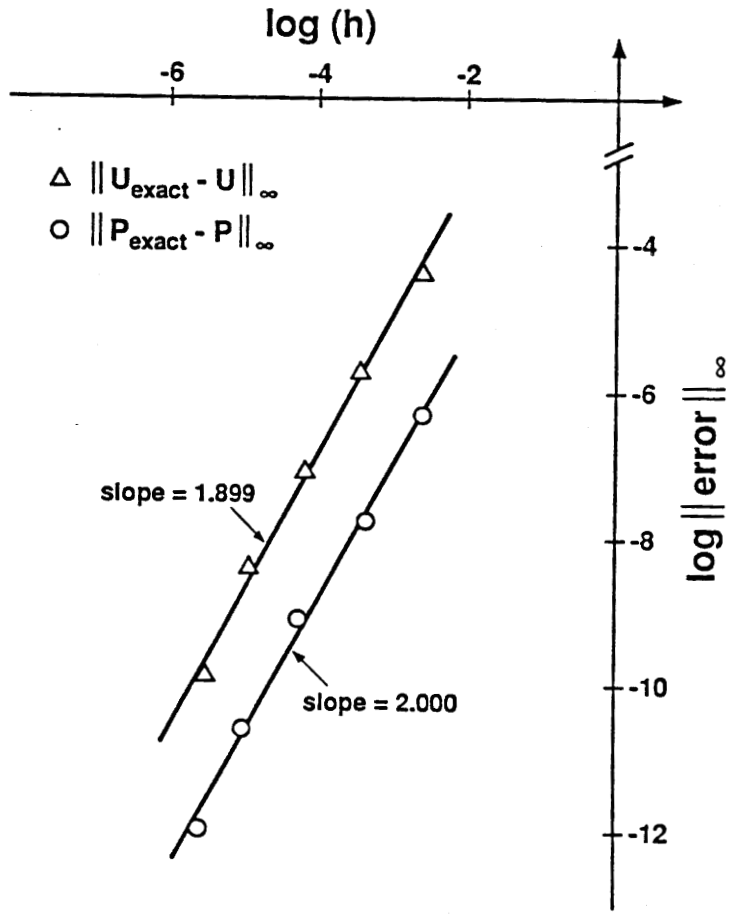


Figure 2. Convergence plot for the mixed finite-element scheme for Poisson's equation, using lowest-order Raviart-Thomas trial spaces. The plots demonstrate the rate of decrease in the nodal errors as $h \to 0$.

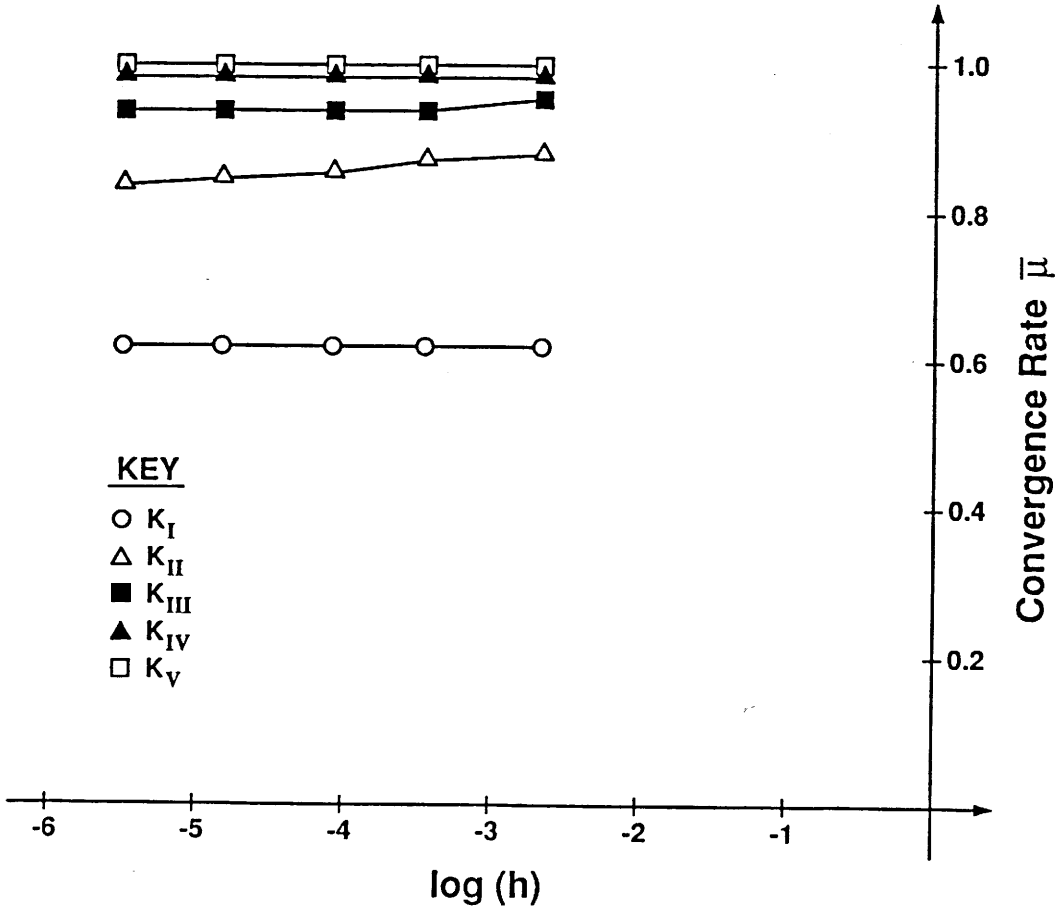


Figure 3. Rate of convergence $\bar{\mu}$ versus grid mesh h for Algorithm 2, using the various choices of coefficient $K(x, y)$.

Appendix: Matrix Structure of the Finite-Element Equations

The mixed finite-element equations (2.2) give rise to integral equations having the following forms: For the x-velocity equation,

$$\int_\Omega \left(K^{-1} u_h^x \, \phi_{i,j}^x + p_h \frac{\partial \phi_{i,j}^x}{\partial x} \right) dx\, dy = 0, \quad i = 0, \ldots, m; \; j = 1, \ldots, n.$$

For the y-velocity equation,

$$\int_\Omega \left(K^{-1} u_h^y \, \phi_{i,j}^y + p_h \frac{\partial \phi_{i,j}^y}{\partial y} \right) dx\, dy = 0, \quad i = 1, \ldots, m; \; j = 0, \ldots, n.$$

For the mass balance,

$$\int_\Omega \left(\frac{\partial u_h^x}{\partial x} + \frac{\partial u_h^y}{\partial y} + f \right) \psi_{i,j} dx\, dy = 0, \quad i = 1, \ldots, m; \; j = 1, \ldots, n.$$

The following integrals appearing in these expressions involve no spatially varying coefficients and hence are easy to compute using the bases for $\mathbf{Q}_h$ and V_h:

$$\int_\Omega p_h \frac{\partial \phi_{i,j}^x}{\partial x} dx\, dy, \int_\Omega p_h \frac{\partial \phi_{i,j}^y}{\partial y} dx\, dy, \int_\Omega \frac{\partial u_h^x}{\partial x} \psi_{i,j} dx\, dy, \int_\Omega \frac{\partial u_h^y}{\partial y} \psi_{i,j} dx\, dy.$$

However, the remaining integrals involve the spatially varying functions $K^{-1}(x,y)$ and $f(x,y)$. We compute these integrals using the mean value theorem for integrals ([9], pp. 184-185) as follows: Since K^{-1} is bounded and integrable on each cell $\overline{\Omega}_{i,j}$, there exist numbers $T_{i,j}^{\mathrm{I}}, T_{i,j}^{\mathrm{II}}, T_{i,j}^{\mathrm{III}}$ such that

$$\int_\Omega K^{-1} \, \phi_{s,t}^x \, \phi_{i,j}^x \, dx\, dy = \begin{cases} T_{i,j}^{\mathrm{II}}/6, & t = j, \; s = i-1; \\ \left(T_{i,j}^{\mathrm{I}} + T_{i+1,j}^{\mathrm{III}} \right)/3, & t = j, \; s = i; \\ T_{i+1,j}^{\mathrm{II}}/6, & t = j, \; s = i+1 \, . \end{cases}$$

Here, $T_{i,j}^{\mathrm{I}}/[(x_i - x_{i-1})(y_j - y_{j-1})]$ is a number lying between the upper and lower bounds of K^{-1} on the cell $\overline{\Omega}_{i,j}$, and similarly for $T_{i,j}^{\mathrm{II}}$ and $T_{i,j}^{\mathrm{III}}$. Analogous calculations show that

$$\int_\Omega K^{-1} \, \phi_{s,t}^y \, \phi_{i,j}^y \, dx\, dy = \begin{cases} T_{i,j}^{\mathrm{V}}/6, & t = j-1, \; s = i; \\ \left(T_{i,j}^{\mathrm{IV}} + T_{i,j+1}^{\mathrm{VI}} \right)/3, & t = j, \; s = i; \\ T_{i,j+1}^{\mathrm{V}}/6, & t = j+1, \; s = i \, . \end{cases}$$

The calculations of $\int_\Omega f \, \psi_{i,j} \, dx\, dy$ can proceed similarly.

Now let us adopt the following orderings for the vectors of unknown nodal coefficients:

$$U^x = \begin{bmatrix} U^x_{0,1} \\ \vdots \\ U^x_{m,1} \\ \vdots \\ U^x_{0,n} \\ \vdots \\ U^x_{m,n} \end{bmatrix}, \; U^y = \begin{bmatrix} U^y_{1,0} \\ \vdots \\ U^y_{1,n} \\ \vdots \\ U^y_{m,0} \\ \vdots \\ U^y_{m,n} \end{bmatrix}, \; P = \begin{bmatrix} P_{1,1} \\ \vdots \\ P_{m,1} \\ \vdots \\ P_{1,n} \\ \vdots \\ P_{m,n} \end{bmatrix}.$$

Then the entire algebraic system arising from Equations (2.2) has the structure

$$\begin{bmatrix} A^x & 0 & N^x \\ 0 & A^y & N^y \\ (N^x)^T & (N^y)^T & 0 \end{bmatrix} \begin{bmatrix} U^x \\ U^y \\ P \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix}.$$

Here,

$$A^x = \begin{bmatrix} A^x_1 & & \\ & \ddots & \\ & & A^x_n \end{bmatrix} \in \mathbb{R}^{(m+1)n \times (m+1)n},$$

where each block $A^x_j \in \mathbb{R}^{(m+1)\times(m+1)}$ has the tridiagonal structure

$$A^x_j = \frac{1}{6} \begin{bmatrix} 2T^{\mathrm{III}}_{1,j} & T^{\mathrm{II}}_{1,j} & & & \\ T^{\mathrm{II}}_{1,j} & 2(T^{\mathrm{III}}_{1,j} + T^{\mathrm{I}}_{2,j}) & T^{\mathrm{II}}_{2,j} & & \\ & & \ddots & & \\ & & & T^{\mathrm{II}}_{m,j} & 2T^{\mathrm{III}}_{m,j} \end{bmatrix}.$$

Similarly,

$$A^y = \begin{bmatrix} A^y_1 & & \\ & \ddots & \\ & & A^y_m \end{bmatrix} \in \mathbb{R}^{m(n+1)\times m(n+1)},$$

where each block $A^y_i \in \mathbb{R}^{(n+1)\times(n+1)}$ has the tridiagonal form

$$A^y_i = \frac{1}{6} \begin{bmatrix} 2T^{\mathrm{IV}}_{i,1} & T^{\mathrm{V}}_{i,1} & & & \\ T^{\mathrm{V}}_{i,1} & 2(T^{VI}_{i,1} + T^{\mathrm{IV}}_{i,2}) & T^{\mathrm{V}}_{i,2} & & \\ & & \ddots & & \\ & & & T^{\mathrm{V}}_{i,n} & 2T^{\mathrm{VI}}_{i,n} \end{bmatrix}.$$

Finally, the two "difference" matrices N^x and N^y have the following structures:

$$N^x = \begin{bmatrix} N^x_1 & & \\ & \ddots & \\ & & N^x_n \end{bmatrix} \in \mathbb{R}^{n(m+1)\times nm},$$

where

$$N_j^x = (y_j - y_{j-1}) \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & & 1 \\ & & & & -1 \end{bmatrix} \in \mathbb{R}^{(m+1)\times m},$$

while

$$N^y = \begin{bmatrix} N_{1,1}^y & \dots & N_{1,n}^y \\ \vdots & & \\ N_{m,1}^y & \dots & N_{m,n}^y \end{bmatrix} \in \mathbb{R}^{(n+1)m\times nm},$$

where

$$N_{i,j}^y = (x_i - x_{i-1}) \begin{bmatrix} & \vdots & \\ \cdots & 1 & \cdots \\ \cdots & -1 & \cdots \\ & \vdots & \end{bmatrix} \begin{matrix} \\ \leftarrow \text{ row } j \\ \leftarrow \text{ row } j+1 \\ \\ \uparrow \\ \text{column } i \end{matrix} \in \mathbb{R}^{(n+1)\times m}.$$

FINITE ELEMENT ANALYSIS IN FLUIDS

Proceedings of the Seventh International Conference on Finite Element Methods in Flow Problems

APRIL 3 – 7, 1989

The University of Alabama in Huntsville
Huntsville, Alabama

T. J. Chung, and Gerald R. Karr, Editors

UAH PRESS
DEPARTMENT OF MECHANICAL ENGINEERING
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE
HUNTSVILLE, AL 35899

MATHEMATICS OF FINITE ELEMENTS

PARALLEL COMPUTING SPEEDUPS FOR ALTERNATING DIRECTION COLLOCATION

Mark C. Curran and Myron B. Allen III
Department of Mathematics, University of Wyoming
Laramie, WY 82070 U.S.A.

ABSTRACT

We apply finite-element collocation to the two-dimensional advection-diffusion equation. Collocation offers savings over other finite-element techniques in that matrix elements are found by point evaluations rather than integrations. Additional computer time and storage is saved by application of an alternating direction process, which allows a multidimensional problem to be solved as a sequence of one-dimensional problems. Since these one-dimensional problems are independent, the speed of the method is enhanced further through use of a parallel computing architecture.

1. INTRODUCTION

Alternating direction (AD) methods have been formulated for the numerical solution of partial differential equations since their introduction in 1955 by Peaceman and Rachford [1]. In 1970 Douglas and Dupont [2] developed the alternating direction Galerkin method. More recently, the alternating direction collocation (ADC) method has appeared in several formulations by Bangia et al. [3], Chang and Finlayson [4], Hayes [5], Celia et al. [6], Celia [7], and Celia and Pinder [8].

We examine Celia's ADC for the two-dimensional advection-diffusion equation. Of special interest here is the amenability of the procedure to implementation on parallel-architecture computers. The paper has the following structure: Section 2 briefly reviews finite-element collocation using a bicubic Hermite basis; Section 3 discusses the AD method applied to collocation; Section 4 concludes the paper with an examination of the method's performance on a parallel computer.

2. REVIEW OF FINITE-ELEMENT COLLOCATION

Consider the following problem posed on the spatial domain $\Omega = (a,b) \times (c,d)$:

$$
\begin{aligned}
&(a) \quad \partial_t u + \mathbf{v} \cdot \nabla u - \nabla \cdot (D \nabla u) = 0, \quad (x,y,t) \in \Omega \times (0,\infty), \\
&(b) \quad u(x,y,0) = u_I(x,y), \quad (x,y) \in \Omega, \\
&(c) \quad u(x,y,t) = u_B(x,y,t), \quad (x,y) \in \partial\Omega, \; t \geq 0.
\end{aligned} \tag{1}
$$

Here $\mathbf{v} = \mathbf{v}(x,y)$ represents fluid velocity; $D = D(x,y)$ is a diffusion coefficient, and $u = u(x,y,t)$ represents solute concentration. We apply finite-element collocation to the following semidiscrete analog:

$$u^{n+1} - u^n + k[\mathbf{v}\cdot\nabla u^{n+\theta} - \nabla\cdot(D\nabla u^{n+\theta})] = 0, \tag{2}$$

where integer superscripts indicate time level, $(\cdot)^{n+\theta} \equiv \theta(\cdot)^{n+1} + (1-\theta)(\cdot)^n$, $0 \le \theta \le 1$, and k signifies the time step.

We begin by establishing a grid on Ω. Let $\Delta_x = \{x_i = a + ih_x, i = 0,\dots,N_x\}$ and $\Delta_y = \{y_j = c + jh_y, j = 0,\dots,N_y\}$, where $h_x = (b-a)/N_x$ and $h_y = (c-d)/N_y$. The Hermite piecewise cubics on these grids are

$$M_1^3(\Delta_x) = \{f \in C^1(\bar{\Omega}) \mid f|_{[x_{i-1},x_i]} \text{ is cubic, } i = 1,\dots,N_x\},$$

and similarly for $M_1^3(\Delta_y)$. As Prenter [9] shows, each of these spaces has an interpolating basis $\{h_{0i}, h_{1i}\}_{i=0}^{N_x \text{ or } N_y}$, every element of which has support confined to at most two adjacent subintervals $[x_{i-1},x_i]$ or $[y_{j-1},y_j]$.

At each time level n we compute an approximate solution $\hat{u}^n(x,y)$ belonging to the tensor-product trial space

$$M = \{v \in M_1^3(\Delta_x) \otimes M_1^3(\Delta_y) \mid v(x,y) = u_B(x,y) \text{ for } (x,y) \in \partial\Omega\}.$$

Each function in M obeys the boundary conditions (1c) and has the form

$$\begin{aligned}\hat{u}^n(x,y) \;=\; \sum_{i=0}^{N_x}\sum_{j=0}^{N_y}\Bigg[&\hat{u}^n(x_i,y_j)H_{00ij}(x,y) + \frac{\partial\hat{u}^n}{\partial x}(x_i,y_j)H_{10ij}(x,y)\\ &+\frac{\partial\hat{u}^n}{\partial y}(x_i,y_j)H_{01ij}(x,y) + \frac{\partial^2\hat{u}^n}{\partial x\partial y}(x_i,y_j)H_{11ij}(x,y)\Bigg],\end{aligned}$$

where $H_{\ell mij}(x,y) = h_{\ell i}(x)h_{mj}(y)$. At $t = 0$ we form $\hat{u}^0$ by projecting the initial function u_I onto M. These criteria specify $\hat{u}^0$ completely and determine $4(N_x + N_y + 1)$ of the $4(N_x+1)(N_y+1)$ nodal coefficients for $\hat{u}^1, \hat{u}^2, \dots$.

To determine the remaining $4N_xN_y$ degrees of freedom at each time level $n+1$, we first form the residual

$$R^{n+1} = \hat{u}^{n+1} - \hat{u}^n + k\left[\mathbf{v}\cdot\nabla\hat{u}^{n+\theta} - \nabla\cdot(D\nabla\hat{u}^{n+\theta})\right].$$

We then pick a collection $\{(\bar{x}_1,\bar{y}_1),(\bar{x}_1,\bar{y}_2),\dots,(\bar{x}_{2N_x},\bar{y}_{2N_y})\}$ of collocation points and force $R^{n+1}(\bar{x}_k,\bar{y}_\ell) = 0$ at each. To obtain optimal $O((h_x+h_y)^4)$ error estimates, we choose $\bar{x}_k$ and $\bar{y}_\ell$ to be the two-point Gauss-quadrature abscissae on each subinterval $[x_{i-1},x_i]$ or $[y_{j-1},y_j]$.

3. THE ALTERNATING DIRECTION METHOD

To obtain a matrix that can be factored into AD form, we first perturb Equation (2) by a term that is $O(k^2)$ to get

$$u^{n+1} - u^n + k(\mathcal{L}_x + \mathcal{L}_y)u^{n+\theta} + k^2\theta^2(\mathcal{L}_x\mathcal{L}_y)(u^{n+1} - u^n) = 0, \tag{3}$$

where

$$\mathcal{L}_x = v_x\partial_x - \partial_x(D\partial_x) \text{ and } \mathcal{L}_y = v_y\partial_y - \partial_y(D\partial_y).$$

Rearranging Equation (3) and factoring gives

$$(1 + k\theta\mathcal{L}_y)(1 + k\theta\mathcal{L}_x)(u^{n+1} - u^n) = -k(\mathcal{L}_x + \mathcal{L}_y)u^n.$$

Now we can solve $(1 + k\theta\mathcal{L}_y)z = -k(\mathcal{L}_x + \mathcal{L}_y)u^n$, followed by $(1 + k\theta\mathcal{L}_x)(u^{n+1} - u^n) = z$.

When we substitute Hermite bicubic trial functions for $\hat{u}$, we get a matrix equation $K\mathbf{u}^{n+1} = \mathbf{r}^n$, where $\mathbf{u}^{n+1}$ is the vector of time increments for the unknown nodal coefficients of $\hat{u}$. Consider a typical entry of K:

$$\left\{\left[1 + k\theta(\mathcal{L}_x + \mathcal{L}_y) + k^2\theta^2(\mathcal{L}_x\mathcal{L}_y)\right]H_\sigma\right\}(\bar{x}_k,\bar{y}_\ell), \tag{4}$$

where H_σ is shorthand for some basis function $H_{\ell mij}$. Each $H_\sigma(x,y) = h_\alpha(x)h_\beta(y)$, with $\alpha = (i,r)$ and $\beta = (j,s)$, so we can expand the expression (4) and factor it to get

$$[h_\alpha(\bar{x}_k) + k\theta(\mathcal{L}_x h_\alpha)(\bar{x}_k)]\cdot[h_\beta(\bar{y}_\ell) + k\theta(\mathcal{L}_y h_\beta)(\bar{y}_\ell)].$$

If we number the nodes along the lines $x = \bar{x}_k$, we can use this observation to factor the $4N_xN_y \times 4N_xN_y$ matrix K as follows:

$$K = Y\cdot X = \begin{bmatrix} Y_{1,1} & & \\ & \ddots & \\ & & Y_{2N_x,2N_x}\end{bmatrix}\cdot\begin{bmatrix} X_{1,1} & \cdots & X_{1,2N_y}\\ \vdots & & \vdots\\ X_{2N_y,1} & \cdots & X_{2N_y,2N_y}\end{bmatrix}.$$

Each $2N_y \times 2N_y$ block $Y_{j,j}$ has the five-band structure of a one-dimensional collocation matrix, and its entries depend only on the y-coordinates of collocation points.

We can solve the matrix equation $K\mathbf{u}^{n+1} = \mathbf{r}^n$ by the following procedure.

1. Order the nodes vertically and solve $Yz = \mathbf{r}^n$ by solving the independent problems $Y_{j,j}z_j = \mathbf{r}_j^n$, $j = 1,\dots,2N_x$.
2. Reorder the nodes horizontally to convert z to z^*. This operation transforms X to a block-diagonal form X^* whose blocks $X_{i,i}^*$ have one-dimensional structure.
3. Solve $X^*\mathbf{u}^{n+1} = z^*$ by solving the independent systems $X_{i,i}^*\mathbf{u}_i^{n+1} = z_i^*$, $i = 1,\dots,2N_y$.

Each of the "one-dimensional" systems in steps 1 and 3 is independent of any other. Therefore these steps can be done concurrently.

4. IMPLEMENTATION ON A PARALLEL COMPUTER

We have implemented ADC on an Alliant FX/8 parallel processing computer. The Alliant is an eight-processor, shared-memory machine with optimization capability for both concurrent and vector programming. The machine allows users to control concurrency within a Fortran code through the use of compiler directives. The following is a description of the code outlined in Steps 1–3 of Section 3. The compiler directives themselves begin with the flag CVD$ starting in the first column of code.

```
      Initialize û⁰, set n = 0
      Begin time level n + 1
CVD$L   CNCALL (Compiler directive to permit the concurrent execution of the following loop
                containing a reference to an external procedure.)
        DO for each j = 1,...,2N_x
          CALL YSWEEP (Constructs the system Y_j,j x_j = r_j^n, solves it and saves the results.)
        END DO
        CALL RENUM (Reorders x to get x*)
CVD$L   CNCALL
        DO for each i = 1,...,2N_y
          CALL XSWEEP (constructs the system X_i^*,i u_i^n+1 = z_i^*, solves it and updates the
                       appropriate coefficients of û to time level n + 1.)
        END DO
      End time step
        ⋮
CVD$R NOCONCUR (Directive to supress concurrency until the end of the routine.)
      SUBROUTINE YSWEEP
CVD$R NOCONCUR
      SUBROUTINE XSWEEP
```

One measure of how well the algorithm makes use of the machine's parallel capabilities is the *speedup*. Speedup for n processors is the ratio of the time needed by one processor to the time used by n processors to perform a set of tasks in parallel. For a perfectly parallel algorithm requiring no overhead to monitor and schedule the various processors, the speedup for n processors would be n. Figure 1 shows the speedup curve for this algorithm, excluding initialization. The speedup for eight processors is 7.27. Clearly, ADC makes very good use of the shared-memory parallel architecture.

To confirm that ADC gives useful approximations, Figures 2 and 3 show solution plots for two different problems. Figure 2 shows the results of a rotating plume problem on $\Omega = (-1,1) \times (-1,1)$, with $N_x = N_y = 40$ and $k = 0.004$. Here, $\mathbf{v} = 2\pi(-y,x)$ is a circular velocity field, $D = 0$, and the initial concentration plume $u_I(x,y)$ is a "Gauss hill" with center at $(0,-0.6)$ and standard deviation $\sigma = 0.066$. Figure 3 displays the results of an advection-diffusion problem on $\Omega = (0,1) \times (0,1)$, with $N_x = N_y = 20$ and $k = 0.004$. The diffusion coefficient is $D = 0.00385$, and $\mathbf{v}(x,y) = 2e^{xy}(-y,x)$. Here, u_I is a "Gauss hill" with $\sigma = 0.05$ centered at $(0.75, 0.25)$. In both problems the global error is less than $.02\|u\|_\infty$.

ACKNOWLEDGMENTS

The Wyoming Water Research Center supported this work. We also received support from NSF grant RII-8610680 and ONR contract 0014-88-K-0370.

REFERENCES

1. Peaceman, D.W. and H.H. Rachford, "The Numerical Solution of Parabolic and Elliptic Equations," *SIAM J.*, *3*, 28–41 (1955).

2. Douglas, J., Jr. and T. Dupont, "Alternating-Direction Galerkin Methods on Re angles," in *Numerical Solution of Partial Differential Equations, Vol. 2*, Synspa 1970, B. Hubbard, Ed., Academic, New York, 1971, pp. 133–214.
3. Bangia, V.K., C. Bennett, and A. Reynolds, "Alternating Direction Collocation f Simulating Reservoir Performance," presented at the 53rd Annual Fall Conferen Society of Petroleum Engineers, Houston, 1978.
4. Chang, P.W. and B.A. Finlayson, "Orthogonal Collocation on Finite Elements f Elliptic Equations," *Math. Comp. Simulation*, 83–92, (1978).
5. Hayes, L.J. "An Alternating-Direction Collocation Method for Finite Element A proximations on Rectangles," *Comput. Math. Appl.*, *6*, 45–50, (1980).
6. Celia, M.A., G.F. Pinder, and L.J. Hayes, "Alternating Direction Collocation Si ulation of the Transport Equation," *Proceedings Third Int. Conf. Finite Eleme in Water Resources*, S.Y. Wang *et al.*, Eds., University of Mississippi, Oxford, M 1980, pp. 3.36–3.48.
7. Celia, M.A., *Collocation on Deformed Finite Elements and Alternating Directi Collocation Methods*, Ph.D. Dissertation, Princeton University, 1983.
8. M.A. Celia and G.F. Pinder, "Generalized Alternating-Direction Collocation Met ods for Parabolic Equations: 1. Spatially Varying Coefficients," (1984).
9. Prenter, P.M., *Splines and Variational Methods*, New York: Wiley, 1975, Chapter

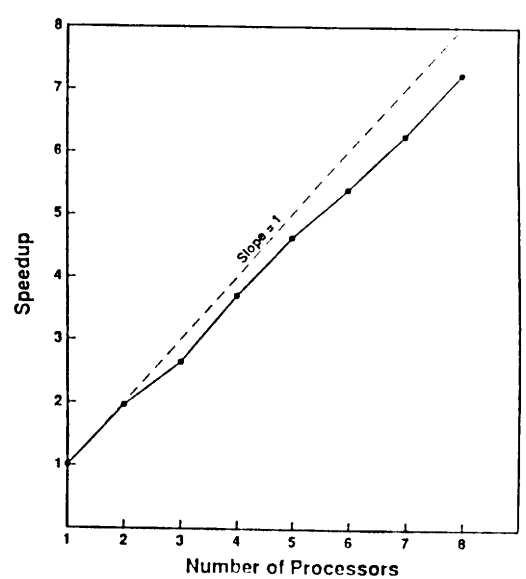


Figure 1. Speedup curve for ADC using the Alliant FX/8 shared-memory architecture.

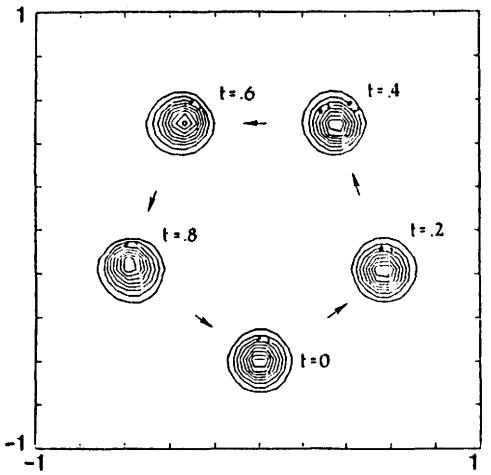


Figure 2. Concentration contours for the purely advective rotating plume problem at various time levels. Contour interval is 0.1.

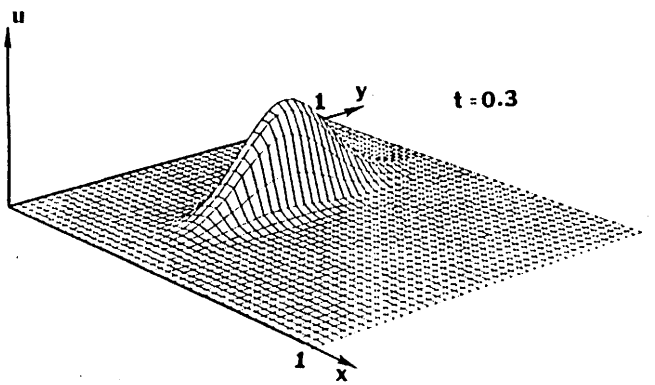


Figure 3. Plot of concentration distribution at $t = 0.3$ for an advection-diffusion problem with potential flow.

Acknowledgement is made to Y. Lecointe and J. Piquet for the use of Figure 5 on page 25, which appears on the front cover of this book.

APPLICATIONS OF SUPERCOMPUTERS IN ENGINEERING:

FLUID FLOW AND STRESS ANALYSIS APPLICATIONS

Proceedings of the first International Conference, Southampton, UK, September 1989

Edited by

C.A. Brebbia
A. Peters

ELSEVIER
Amsterdam - Oxford - New York - Tokyo 1989

Co-published with

COMPUTATIONAL MECHANICS PUBLICATIONS
Southampton - Boston 1989

A Parallel Collocation Based Algorithm for the Generalized Transport Equation

J.F. Guarnaccia and G.F. Pinder
Department of Civil Engineering, University of Vermont
Burlington, VT 05405-0156, USA

INTRODUCTION

The solution of the generalized transport equation in porous media can be a computationally intensive task requiring large amounts of computer time. A worst case scenario involves multiphase problems which require the simultaneous solution of coupled nonlinear equations, as well as fine time and space discretizations to match stability and accuracy constraints. The turnaround time for a given simulation on serial computers can be on the order of hours or days depending on model size. For the practicing engineer, long turnaround times during the calibration phase of model development can limit its application. As a result, a new numerical algorithm has been developed to speed up transport simulations by implementing parallel processing computer technology.

Conceptually we want a method which exhibits high accuracy in time and space, is amenable to a parallel processing environment, and is easy to implement. To this end, the proposed method employs a combination of several numerical techniques. To transform the governing equation into a set of algebraic equations we employ an implicit backward difference approximation for the time derivative and the collocation finite element method to approximate the space derivatives. Even though an implicit finite differencing in time is only first order accurate, it results in a highly stable solution scheme.

We choose to use the collocation finite element method for several reasons. First, the method has been successfully applied to a wide range of engineering problems including problems in porous media physics (Finlayson, 1972, Frind and Pinder, 1979, among others). Second, as a method of weighted residuals, collocation employs the displaced Dirac Delta function as its trial function. This results in driving the error to zero at specified points in the domain (called collocation points), and as a result, unlike the finite-element method, no formal integrations need be performed. Thus, system matrix assembly is analogous to the finite difference method. Third, because of continuity requirements, the

approximating function of interest is cast in the C^1 continuous hermite cubic basis. Given this, if we choose as the collocation points the Gauss points, the method exhibits fourth order spatial accuracy. Fourth, boundary conditions are easily incorporated into the formulation.

While the above formulation provides a method which emphasizes accuracy and ease of implementation, to cast this into a parallel algorithm framework we will employ an idea based on alternation direction (AD) techniques. AD methods are characterized by the solution of multi-dimensional problems by a series of effectively one-dimensional solutions. The motivation for the development of AD methods for use on serial computers has been the reduced matrix storage requirements and reduced execution time (Celia, et. al., 1980, Hayes, et. al., 1981, Celia and Pinder, 1985). AD schemes which allow for independent processing of each resulting one-dimensional problem have been implemented on parallel processing computers of various architectures with good results (Johnsson, et. al., 1985, Hockney and Jesshope, 1988).

Classically, AD methods achieve the spatial split by factoring the space operator of the partial differential equation into its spatial components (Douglas and Gunn, 1964). As a result, a problem inherent in all AD methods is their inability to directly accommodate cross derivative terms in the space operator (Mckee and Mitchell, 1970). Cross derivative terms arise in contaminant transport problems in the form of permeability and dispersion tensor coefficients, and are prominent when flow is not coincident with one of the axes of the coordinate system used to discretize the problem domain. Because cross derivative terms require milti-directional information, operator splitting schemes can accommodate them explicitly only by lagging their influence by a time step or iteration cycle. Therefore, the accuracy of these methods is dependent, to some degree, on the component of flow and transport crossing coordinate lines. Because of this dependency on the orientation of the grid with respect to the flow path, some AD schemes are susceptible to grid orientation effects (Glimm, et. al., 1981). In other words, the solution of the problem is a function of the relative orientation of the grid to the flow path.

The AD algorithm to be presented herein, called Parallel Alternating Direction Collocation (PADC), does not require the space operator to be factored. Instead, the spatial split is achieved by modifying the system matrix.

METHOD DEVELOPMENT

Consider a general linear transport equation with constant coefficients:

$$\frac{\partial u}{\partial t} + L u = 0, \quad (x, y, t) \in \Omega \times (0, T) \qquad (1)$$

intersection of element boundaries) at which we define an undetermined coefficient vector (u) and a basis function vector (φ). Note that the superscript denotes a transpose.

The vector of space dependent basis functions, $\varphi_i(x, y) \equiv [\varphi^{00}, \varphi^{10}, \varphi^{01}, \varphi^{11}]_i^T$, are piecewise Hermite cubic polynomials (Lapidus and Pinder, 1982). Both the functions and their first derivatives are continuous across element boundaries (C^1 continuity). The time dependent undetermined coefficients in equation (2), $\mathbf{u}_i^T(t) = [\mathbf{u}, \mathbf{u}_{,x}, \mathbf{u}_{,y}, \mathbf{u}_{,xy}]_i$, are the nodal values of $\hat{u}$ and its x, y, and cross derivatives respectively.

The task is to solve equation (1) numerically given that it is subject to prescribed initial and boundary conditions. The procedure we will employ requires that we first derive a fully 2-D Collocation approximation and then rewrite it in an AD form.

Let us first approximate the time derivative by using an implicit backward difference approximation:

$$\frac{u^{n+1} - u^n}{\Delta t} + L\,u^{n+1} = 0 \qquad (3)$$

where Δt is the time discretization and the superscript, n, denotes the time level of solution (ie. $t=n\Delta t$). This approximation results in a scheme which is locally first order accurate.

The approximate function $\hat{u}(x,y,t)$ (equation 2) is now substituted for u(x,y,t) in equation (3) and, since it is an approximation, it will not satisfy the equation exactly leaving a residual, R (x,y,t) :

$$\frac{\hat{u}^{n+1} - \hat{u}^n}{\Delta t} + L\,\hat{u}^{n+1} = R^{n+1} \qquad (4)$$

This residual is driven to zero in a weighted average sense by taking the inner product of R with the displaced Dirac Delta function. This is equivalent to driving the error to zero at specified points in the domain which are denoted as collocation points. Thus no formal integrations are required, and the system matrix generation is computationally analogous to the finite difference method in that we write equations at points in the domain. If we substitute equation (2) into (4) and evaluate it at a select set of collocation points, we obtain a system of linear simultaneous equations which can be written as:

$$\sum_{i=1}^{N} \frac{\left[u_i^{n+1} - u_i^n\right]^T}{\Delta t} \varphi_i(x_k, y_k) + \left[u_i^{n+1}\right]^T L\,\varphi_i(x_k, y_k) = 0 \qquad (5)$$

$$k = 1, 2, \ldots, 4 \times M;$$

where 'L' is the space operator defined as:

$$L(\bullet) = v_x \frac{\partial(\bullet)}{\partial x} + v_y \frac{\partial(\bullet)}{\partial y} - D_{xx} \frac{\partial^2(\bullet)}{\partial x^2} - 2D_{xy} \frac{\partial^2(\bullet)}{\partial x \partial y} - D_{yy} \frac{\partial^2(\bullet)}{\partial y^2}$$

and, $u(x,y,t)$ = concentration (mass per unit volume),
$v_x, v_y\,(x,y,t)$ = velocity vector compponents,
$D_{xx}, D_{xy}, D_{yy}\,(x,y,t)$ = symmetric dispersion tensor components.

with prescribed initial and boundary conditions defined on a rectangular domain (Ω).

Dispersion is a symmetric tensor that combines the coefficient of molecular diffusion with mechanical dispersion. Bear (1979) writes the mechanical dispersion tensor as:

$$D_{xx} = \frac{a_l v_x^2}{|\bar{v}|} + \frac{a_t v_y^2}{|\bar{v}|} + D_m$$

$$D_{yy} = \frac{a_l v_y^2}{|\bar{v}|} + \frac{a_t v_x^2}{|\bar{v}|} + D_m$$

$$D_{xy} = D_{yx} = (a_l - a_t) \frac{v_x v_y}{|\bar{v}|}$$

where: D_m = coefficient of molecular diffusion,
a_l = longitudinal dispersivity of the porous medium (in the direction of mean flow),
a_t = transverse dispersivity of the porous medium (in the direction perpendicular to mean flow),
$|\bar{v}|$ = mean velocity magnitude.

It can be easily seen that if one of the coordinate axes of the domain is not coincident with the mean flow path, then $D_{xy} \neq 0$. It is when these terms represent a major transport mechanism that classical AD methods have difficulty capturing the transport physics.

Approximating the function of interest, $u(x,y,t)$, by $\hat{u}(x,y,t)$, a linear combination of weighted basis functions $\varphi_i(x,y)$, one obtains:

$$u(x,y,t) \approx \hat{u}(x,y,t) = \sum_{i=1}^{N} u_i^T(t)\, \varphi_i(x,y) \qquad (2)$$

where N is the number of nodes and (i) is the nodal index (identified with the

where M is the number of elements, and (x_k, y_k) is the location of a collocation point. Because, in this analysis, we have chosen the location of the collocation points to be the zeros of the Legendre polynomials (Gauss points) there are four collocation points per element. When these points are chosen, the method is called "Orthogonal Collocation", and Prenter (1976) has shown it to yield fourth order spatial accuracy.

The matrix form of equation (5) is given in Figure (1a), where b contains information regarding boundary conditions, and

$$a_{ij} = \varphi(x_i, y_i) + L\,\varphi(x_i, y_i)$$

$$b_{ij} = \varphi(x_i, y_i)$$

where i represents the collocation point and j the nodal degree of freedom. Because we have the function and its derivatives defined at the nodes, boundary conditions are easily incorporated by direct specification of the undetermined coefficient at the boundary nodes.

If we view the finite element grid as a series of discrete rows or columns as in Figure 1b, and number our equations (collocation points) and unknowns (nodal degrees of freedom) accordingly, the system matrix A takes on a regular block structure. An example of a hroizontal sweep numbering scheme and resulting matrix structure is shown in Figure 2. This structure is equivalent to a system of tri-diagonal sub-matricies, shown in figure (1c) as C_{ij}, B_{ij}. The subscripts of the sub-matricies are given as follows:

- i - denotes the row or column along which the collocation equations are written,
- j - denotes the row or column of nodal unknowns associated with those equations.

For example, given a row-wise numbering, the sub-matrix B_{12} represents those equations written at collocation points along row 1, involving the nodal unknowns along row 2.

The AD spatial split is achieved by moving the off-diagonal sub-matricies (B_{ij}, $i \neq j$) to the right hand side by projecting the unknowns associated with them to the new time level. This projection is defined as:

$$u^{n+1} \approx u^{n+1^*} = \gamma\, u^n + (1 - \gamma)\, u^{n-1}$$

where: $\gamma \equiv$ *projection parameter*, $1 \leq \gamma \leq 2$

It should be noted that if $\gamma = 2$ the projection is second order accurate in time, otherwise it is first order accurate. Figure (1d) shows the system matrix structure after the projection has been made. Note that the system matrix is block diagonal. Each block represents the coefficients in a row or a column of nodal unknowns. The system has been spatially decoupled and all the sub-matricies can be solved concurrently, thereby, reducing a two-dimensional

where 'L' is the space operator defined as:

$$L(\bullet) = v_x \frac{\partial(\bullet)}{\partial x} + v_y \frac{\partial(\bullet)}{\partial y} - D_{xx} \frac{\partial^2(\bullet)}{\partial x^2} - 2D_{xy} \frac{\partial^2(\bullet)}{\partial x \partial y} - D_{yy} \frac{\partial^2(\bullet)}{\partial y^2}$$

and, $u(x,y,t)$ = concentration (mass per unit volume),
$v_x, v_y\,(x,y,t)$ = velocity vector compponents,
$D_{xx}, D_{xy}, D_{yy}\,(x,y,t)$ = symmetric dispersion tensor components.

with prescribed initial and boundary conditions defined on a rectangular domain (Ω).

Dispersion is a symmetric tensor that combines the coefficient of molecular diffusion with mechanical dispersion. Bear (1979) writes the mechanical dispersion tensor as:

$$D_{xx} = \frac{a_l v_x^2}{|\bar{v}|} + \frac{a_t v_y^2}{|\bar{v}|} + D_m$$

$$D_{yy} = \frac{a_l v_y^2}{|\bar{v}|} + \frac{a_t v_x^2}{|\bar{v}|} + D_m$$

$$D_{xy} = D_{yx} = (a_l - a_t) \frac{v_x v_y}{|\bar{v}|}$$

where: D_m = coefficient of molecular diffusion,
a_l = longitudinal dispersivity of the porous medium (in the direction of mean flow),
a_t = transverse dispersivity of the porous medium (in the direction perpendicular to mean flow),
$|\bar{v}|$ = mean velocity magnitude.

It can be easily seen that if one of the coordinate axes of the domain is not coincident with the mean flow path, then $D_{xy} \neq 0$. It is when these terms represent a major transport mechanism that classical AD methods have difficulty capturing the transport physics.

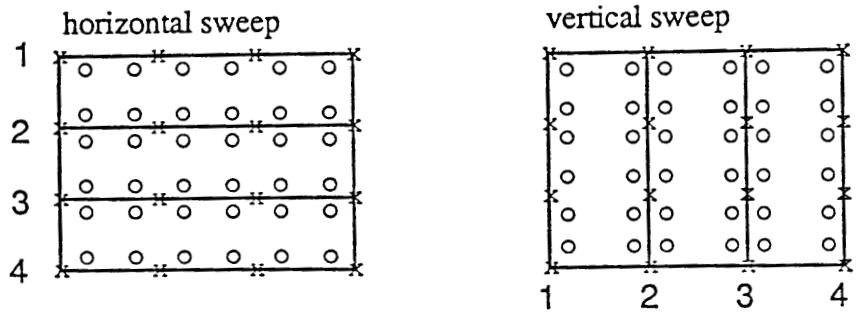
Approximating the function of interest, $u(x,y,t)$, by $\hat{u}(x,y,t)$, a linear combination of weighted basis functions $\varphi_i(x,y)$, one obtains:

$$u(x,y,t) \approx \hat{u}(x,y,t) = \sum_{i=1}^{N} u_i^T(t)\, \varphi_i(x,y) \qquad (2)$$

where N is the number of nodes and (i) is the nodal index (identified with the

a.) $[A]\{u\}^{n+1} = R^n + b$

b.)



c.)

$$\begin{bmatrix} C_{11} & B_{12} & 0 & 0 \\ B_{21} & C_{22} & B_{23} & 0 \\ 0 & B_{32} & C_{33} & B_{34} \\ 0 & 0 & B_{43} & C_{44} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}^{n+1} = R^n + b$$

d.)

$$\begin{bmatrix} C_{11} & 0 & 0 & 0 \\ 0 & C_{22} & 0 & 0 \\ 0 & 0 & C_{33} & 0 \\ 0 & 0 & 0 & C_{44} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}^{n+1} = R^n + b - \begin{bmatrix} 0 & B_{12} & 0 & 0 \\ B_{21} & 0 & B_{23} & 0 \\ 0 & B_{32} & 0 & B_{34} \\ 0 & 0 & B_{43} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}^{(n+1)^*}$$

Figure 1 - A diagrammatic representation of the PADC method for linear systems. (a.) the fully 2-D system where the vector of unknowns **u** at the new time level (n+1) is solved for by using old time level (n) and boundary (b) information. (b.) treat the grid as discrete columns or rows where the x's are nodal locations and the o's are collocation point locations, (c.) given a numbering scheme reflecting (b), the system matrix A is resolved into a tri-diagonal set of sub-matrices C_{ij} and B_{ij}. The subscript i denotes the row or column along which the equations are written (collocation points), and subscript j the row or column of unknowns (degrees of freedom). The structure of sub-matrices C_{ij} and B_{ij} are respectively the same. (d.) the spatial split is achieved by moving the off diagonal sub-matrices B_{ij} to the right hand side by projecting the unknowns associated with them to the new time level (n+1*). Because each sweep biases the boundary conditions differently, a solution to the system is achieved only after an x and a y sweep (complete solution at time level n+2).

problem into a series of one-dimensional problems. This scheme is analogous to the block Jacobi iterative method used to solve simultaneous equations.

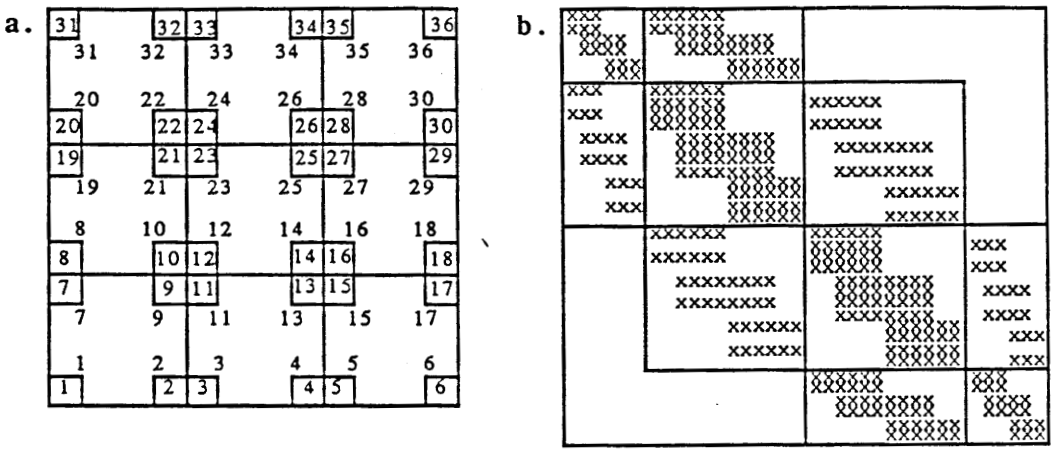


Figure 2 - Part (a), horizontal numbering scheme where the boxed numbers represent the unknowns and the non-boxed numbers represent the collocation points. Part (b), the resulting system matrix structure (36x36) where the rows are the equations associated with each collocation point and the columns are the coefficients (x's) multiplying each unknown. Only non-zero entries are depicted.

Because each grid orientation biases the boundary conditions differently, as well as for stability reasons, one must alternate between horizontal and vertical sweeps as the solution marches through time. Therefore, a complete solution to the system is obtained after two time steps, a horizontal sweep to go from time level 'n' to 'n+1', and a vertical sweep from level 'n+1' to 'n+2'.

The scheme has been shown to to exhibit first order convergence in time and fourth order in space (both at theoretical limits). In addition the method has been shown to have conditional stability properties that assure its utility over the practical range of applications of general transport problems.

PARALLEL IMPLEMENTATION

The Parallel Alternating Direction Collocation (PADC) algorithm set up to solve the two-dimensional linear transport equation was run on an Alliant FX/8 parallel processing computer. This computer is characterized by eight tightly-coupled, identical, processing elements (PE's) which are capable of executing any task and which share a common memory for easy data and instruction

exchange. This parallel architecture is classified as being multiple instruction multiple data (MIMD). In developing an efficient algorithm on a MIMD computer, one must address the computational costs associated with distributing the tasks. These computational costs include (after Hockney and Jesshope 1988):

1.) *Scheduling:* How efficiently are the processors being utilized? One wants to minimize the time a processor has to wait for another processor to finish a task.The *efficiency of scheduling* (E_p) is defined as:

E_p= (one processor execution time) / [(n processor execution time) x n]

where $E_p \leq 1$. E_p decreases with wait time, and typically decreases as 'n' increases. Perfect scheduling is approached when a task is divided into equal *work segments* distributed to an even multiple of the processors.

2.) *Synchronization:* How long can a process run before it needs data from another process such that operations are performed in the correct sequence? One attempts to set algorithm *granularity* (size of the work segments) such that the ratio of data transfer time to process run time is small. The concept of having large *work segments* or *coarse granularity* is well suited for MIMD architectures. The overhead is associated with the time it takes to transfer data.

3.) *Communication:* How long can a process run before it needs data from another process such that operations are performed on the correct data? One wants to minimize the ratio of memory access to arithmetic operations. *Synchronization* costs are a function of *communication* costs.

The Alliant FX/8 hardware design is aimed at minimizing the communication costs associated with parallel algorithms. Each PE has a concurrency control unit which distributes the work among the other PE's and synchronizes the calculation. For example, if the iterations of a DO loop are to be run in parallel, the programmer simply includes a directive just before the DO loop instructing the machine to execute the iterations in parallel while the hardware maintains local data dependencies (each processor may be working on the same vector). Of course, the programmer must still keep a sharp eye on such things as recursive data dependencies (the case where the data generated in one loop iteration is required as input in the next loop).

In order to aid the programmer in algorithm development, the Alliant includes an optimizing compiler which examines loops in the code for parallel potential. The output from this compiler lists those loops which can be parallelized. The programmer is then free to either choose which loops are to be run in parallel based on estimated overhead costs described above or alter the code to highlight additional parallelism. An internal timing routine gives the programmer an idea of where most of the work is being done in the code, and he or she can adjust accordingly. Thus, the programmer interacts with the machine until the desired result is obtained. The parameter that is used to quantify algorithm performance is the *speed-up* (Sp), defined as (Kuck, 1978):

S_p= (one processor execution time) / (n processor execution time)

where $S_p \leq n$ (S_p decreases with increased overhead). Optimal speed-up for an n-processor system would be n.

The PADC algorithm was developed to be highly parallelizable. A high percentage of the computational work involved in obtaining a solution is done in concurrent mode. This is an important point in that significant speedups can only be achieved when at least 90 percent of the computations are done in parallel (Monkhoff, 1984). In addition, there is a theoretical limit on speedup, known as Amdahl's law (Amdahl, 1967), given that the parallel parts of the code take zero time, speed-up is determined by the time required to execute the serial portion of the code. A simple flow chart is presented in Figure 3. Each box represents a set of computations. The computations in heavily lined boxes are done in parallel. For large problems (1000 nodes) these computations represent approximately 99% of the total work. This algorithm exhibits coarse granularity which acts to minimize communication costs by employing long work segments.

To obtain a measure of performance a series of model problems were run each being timed on one to eight processors, yielding a *speed-up curve* (Figure 4). The problem domains were rectangular, and the problems differed only in their aspect ratio. In addition the space discretization was an even multiple of eight. Table 1 presents an average of the S_p and E_p values for the different runs. These results show excellent speedup performance. One aspect of these results is worth detailing. In general E_p decreases with the number of processors utilized; however, in this case E_p for the eight processor run (0.84) is higher than E_p for both the six and seven processor runs (0.83 and 0.81 respectively). This is due to the fact that the grid discretization was an even multiple of eight and thus, scheduling should be better for the eight processor run.

exchange. This parallel architecture is classified as being multiple instruction multiple data (MIMD). In developing an efficient algorithm on a MIMD computer, one must address the computational costs associated with distributing the tasks. These computational costs include (after Hockney and Jesshope 1988):

1.) *Scheduling:* How efficiently are the processors being utilized? One wants to minimize the time a processor has to wait for another processor to finish a task.The *efficiency of scheduling* (E_p) is defined as:

E_p= (one processor execution time) / [(n processor execution time) x n]

where $E_p \leq 1$. E_p decreases with wait time, and typically decreases as 'n' increases. Perfect scheduling is approached when a task is divided into equal *work segments* distributed to an even multiple of the processors.

2.) *Synchronization:* How long can a process run before it needs data from another process such that operations are performed in the correct sequence? One attempts to set algorithm *granularity* (size of the work segments) such that the ratio of data transfer time to process run time is small. The concept of having large *work segments* or *coarse granularity* is well suited for MIMD architectures. The overhead is associated with the time it takes to transfer data.

3.) *Communication:* How long can a process run before it needs data from another process such that operations are performed on the correct data? One wants to minimize the ratio of memory access to arithmetic operations. *Synchronization* costs are a function of *communication* costs.

The Alliant FX/8 hardware design is aimed at minimizing the communication costs associated with parallel algorithms. Each PE has a concurrency control unit which distributes the work among the other PE's and synchronizes the calculation. For example, if the iterations of a DO loop are to be run in parallel, the programmer simply includes a directive just before the DO loop instructing the machine to execute the iterations in parallel while the hardware maintains local data dependencies (each processor may be working on the same vector). Of course, the programmer must still keep a sharp eye on such things as recursive data dependencies (the case where the data generated in one loop iteration is required as input in the next loop).

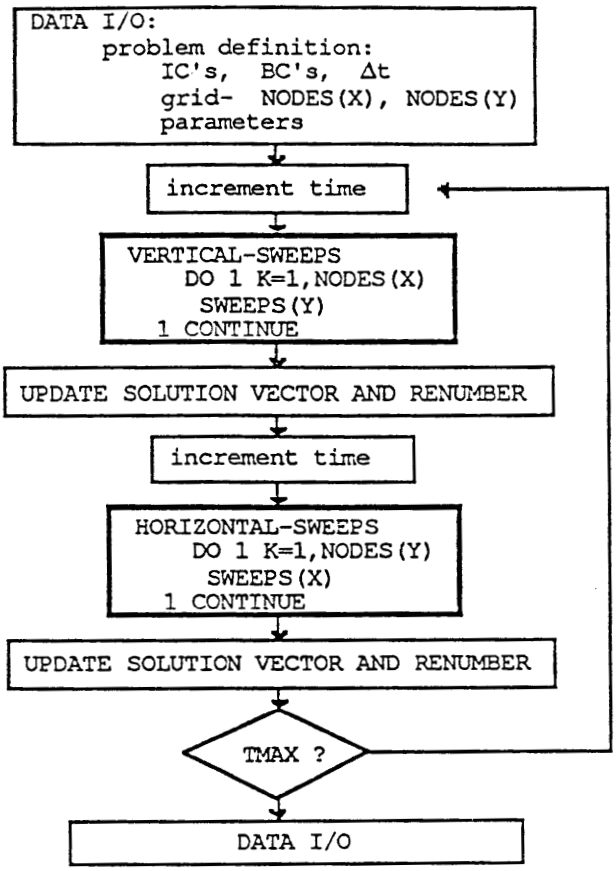


Figure 3 : A simplified flow chart of the PADC linear solution algorithm. Each box represents a set of computations. The heavy lined boxes represent computations done in parallel. For large problems (greater than 1000 nodes) approximately 99% of the computations are done in parallel.

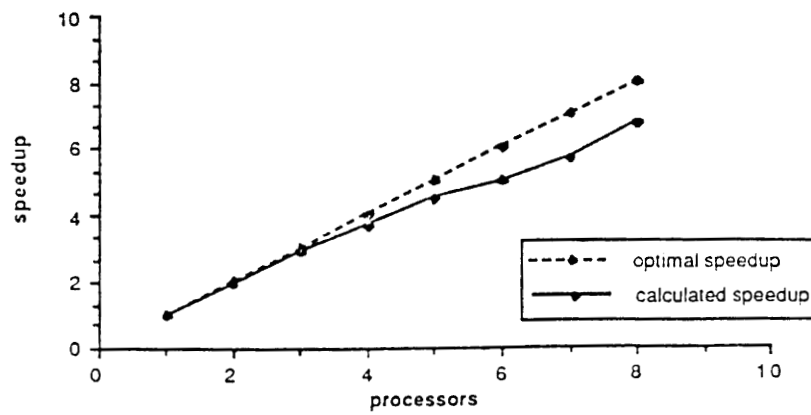


Figure 4 : Speed-up curve for the model problem.

TABLE 1

processors	S_p	E_p
1	1.0	1.0
2	1.98	0.98
3	2.9	0.97
4	3.7	0.93
5	4.5	0.90
6	5.0	0.83
7	5.7	0.81
8	6.7	0.84

ACKNOWLEDGMENT

The authors wish to acknowledg the financial support of the Department of Energy (Grant No. DE-FG02-86ER60453), IBM Corporation, U.S. Environmental Protection Agency assistance I.D. No. CR-814946-01-1, and the Wyoming Water Resources Center. We also wish to acknowledg the use of the computer at the Wyoming Institute for Scientific Computation.

REFERENCES

Amdahl G.M. (1967), "The Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," *AFIPS Conf. Proc. Spring Joint Comput. Conf.*, **30**, pp 483-485.

Bear, J. (1979), *Hydraulics of Groundwater*, Mcgraw-Hill, NY, pp 231-239.

Celia, M.A., G.F. Pinder and L. J. Hayes (1980), "Alternating-Direction Collocation Solution to the Transport Equation," *Proc. Third Int. Conf. Finite Elements in Water Resources*, Wang, et. al. (eds.), Univ of Miss., pp 3.36-3.48.

Celia, M.A. and G.F. Pinder (1985), "An Analysis of Alternating-Direction Methods for Parabolic Equations," *Numerical Solutions for Partial Differential Equations*, **1**, pp 57-70.

Douglas, J., Jr. and J. E. Gunn (1964), "A General Formulation of Alternation Direction Methods, Part 1, Parabolic and Hyperbolic Problems," *Numer. Math.*, **6**, pp 428-453.

Finlayson, B.A. (1972), *The Method of Weighted Residuals and Variational Principles*, Academic Press, New York.

Frind, E.O. and G.F. Pinder (1979), "A collocation Finite Element Method for Potential Problems in Irregular Domains," *Int. J. Num. Meth. Engrg.*, **11**, pp 681-701.

Glimm, J., D. Marchesin and O. McBryan (1981), "Unstable Fingers in Two Phase Flow," *Comm. on Pure and Appl. Math*, 34, pp 53-75.

Hayes, L., G. Pinder and M. Celia (1981), "Alternating-Direction Collocation for Rectangular Regions," *Comp. Meth. Appl. Mech. Engrg.*, 27, pp

265-277.
Hockney, R.W. and C.R. Jesshope (1988), *Parallel Computers 2*, Adam Hilger, Philadelphia, pp 524-550.
Johnsson, S.L., Y. Saad and M.H. Schultz (1985), "Alternating Direction Methods on Multiprocessors," Research Report YALEU/DCS/RR-382, October.
Kuck D. J, (1978), *The Structure of Computers and Computations*. Wiley, New York, p33.
Lapidus, L. and G.F. Pinder (1982), *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley, New York, pp 66-73.
McKee, S. and A.R. Mitchell (1970), "Alternating Direction Methods for Parbolic Equations in Two Space Dimensions with a Mixed Derivative," *The Computer Journal*. 13, No.1, Feb., pp 81-86.
Monkhoff, N. (1984), "Future Computers: Architecture - Parallelism Makes a Strong Bid for Next Generation Computers," *Computer Design*. 23, No.10, September, pp 303-310.
Prenter, P. M. and R. D. Russell (1976), "Orthogonal Collocation for Elliptic Partial Differential Equations," *S.I.A.M. J. Num. Anal.* , 13 , No. 6, pp 923-939.

Proceedings, Groundwater Engineering and Management Conference, Organized by Colorado Water Resources Research Institute and Office of the State Engineer, Denver, CO, February 28-March 1, 1990, pp. 161-170.

HOW AQUIFER HETEROGENEITIES AFFECT NUMERICAL GROUNDWATER MODELS

by Myron B. Allen * and Richard E. Ewing **

1. INTRODUCTION.

One can argue that the nature of heterogeneities in an underground formation is the most influential factor limiting the success of mathematical models of flow or transport in the aquifer. Lack of adequate knowledge of aquifer heterogeneity and the attendant difficulty in assessing the realism of a model's predictions make the modeler's job a frustrating one.

The importance of heterogeneity elicits discomfort among many of us whose research concerns new numerical techniques for groundwater modeling. Part of the uneasiness over heterogeneity arises from a widely shared view of its importance. The valid premise of this view is that difficulties in accurately quantifying underground heterogeneity impose constraints on the accuracy of mathematical models, owing to limitations in the quality of the input data. The argument then proceeds as follows: Since poorly quantified heterogeneity is the dominant source of prediction error in most groundwater models, there is little point in focusing research on improved numerical techniques. After all, even if we use more accurate numerics, the deleterious effects of inadequate input data will still be present, swamping any improvements to be gained through mathematical refinements. The natural conclusion is that research into methods for detecting and characterizing underground heterogeneity have much more potential for improving mathematical models than does research into the numerical techniques themselves.

We offer a different perspective. No one would deny that improved methods for quantifying heterogeneity are crucial to advances in the realism and utility of groundwater models, in accordance with the popular maxim, "garbage in, garbage out." However, we contend that the most commonly used mathematical methods are inadequate to model heterogeneous aquifers. As we review in Section 3, even in the ideal case when the heterogeneities are "perfectly" known, standard methods can perform poorly, suggesting a new adage: "heterogeneity in, garbage out."

---

* Department of Mathematics, University of Wyoming, Laramie, WY 82071.

** Departments of Mathematics, Petroleum Engineering, and Chemical Engineering, University of Wyoming, Laramie, WY 82071.

In a more realistic scenario, where one relies on detailed statistical characterizations of heterogeneous aquifers, existing mathematical techniques are largely inadequate at answering the hydrologist's questions. Here, quantifiable control over the uncertainties in aquifer parameters can fail to yield reasonable control over the reliability of the numerical solution. We illustrate this problem in Section 4. This observation suggests the even more distressing adage: "statistics in, garbage out."

In what follows, we examine these notions and briefly indicate promising avenues for overcoming the difficulties. We hope to affirm the importance of continued research into mathematical techniques used in numerical models of groundwater flow and transport, thereby rebutting the conclusions of the conventional wisdom.

2. GOVERNING EQUATIONS.

To clarify what we mean by aquifer heterogeneity, it is useful to review the governing equations used in groundwater models. We begin with the equations describing groundwater flow. Groundwater obeys a mass balance,

$$S_s \frac{\partial H}{\partial t} + \nabla \cdot \mathbf{v} = Q, \tag{1}$$

where S_s is the specific storage, H is the hydraulic head, $\mathbf{v}$ denotes the superficial velocity, and Q accounts for sources and sinks (Huyakorn and Pinder, 1983, Section 4.2). According to Darcy's law, $\mathbf{v}$ is related to the hydraulic head by the equation

$$\mathbf{v} = -\frac{\rho g k}{\mu} \nabla H = -K \nabla H. \tag{2}$$

Here, ρ is the density of water; g is the gravitational acceleration; k is the permeability of the rock matrix, and μ is the water's dynamic viscosity. Hydrologists typically use the lumped parameter K, called the hydraulic conductivity. In many contexts, Equation (2) is too restrictive, and k (and hence K) must be a tensor to accommodate anisotropies in the aquifer's flow characteristics. This consideration can have practical importance, but it imposes complications that are not essential to our thesis.

Heterogeneity, in this context, refers to spatial variations in the aquifer parameters $S_s(x, y, z)$ and $K(x, y, z)$. For simplicity, we focus on variations in K. A wide array of phenomena associated with the rock's deposition and diagenesis contribute to these variations, which may occur smoothly or

discontinuously. One point that is all too easy to neglect in this connection is that Darcy's law describes the *macroscopic* velocity of fluids, which in some sense represents an average of the velocity in the tortuous, microscopic interstices of the rock that are essentially inaccessible to observation. Thus spatial variations in K reflect what we might call *macroscopic heterogeneity*, as opposed to the microscopic variability in pore geometry that Equations (1) and (2) cannot explicitly model, even in principle.

In formulating numerical models of groundwater flow, people commonly substitute Equation (2) into Equation (1) to get the groundwater flow equation,

$$S_s \frac{\partial H}{\partial t} - \nabla \cdot (K \nabla H) = Q. \tag{3}$$

As we discuss in Section 3, discretizing Equations (1) and (2) separately can yield significant advantages over the usual approaches to discretizing Equation (3).

When the modeler is interested in how a dissolved contaminant moves in a flowing aquifer, it is necessary to solve a transport equation. In the absence of chemical reactions and interphase mass transfer, the equation governing the concentration $c(x, y, z, t)$ of solute has the form

$$\frac{\partial(\phi c)}{\partial t} + \nabla \cdot (\mathbf{v}c) - \nabla \cdot (\phi \mathbf{D} \nabla c) = q. \tag{4}$$

Here, ϕ stands for the porosity of the rock matrix, q accounts for sources and sinks of contaminant, and $\mathbf{v}$ is the velocity computed using a flow model. $\mathbf{D}$ denotes the hydrodynamic dispersion tensor, which is a crude attempt to account for a variety of microscopic phenomena that cause the macroscopically observed concentration to diffuse with respect to the advective field $\mathbf{v}$.

We shall not delve into the controversial physics of $\mathbf{D}$ (see Fried, 1975, Chapter 2, for the standard model). We note, though, that techniques for evaluating $\mathbf{D}$ in actual fieldwork are quite poor and are possibly sensitive to what length scales the measurer identifies as microscopic. Notwithstanding, $\mathbf{D}$ can exhibit macroscopic spatial variations attributable, in the standard model, both to variations in the rock matrix and to variations in $\mathbf{v}$. Thus heterogeneity affects the transport equation (4) through the variability in $\mathbf{v}$ inherited from flow models, through spatial variations in porosity ϕ, and through the intrinsic variability in $\mathbf{D}$.

These equations suffice to illustrate our views on heterogeneity; however, more complicated underground flows have attracted considerable recent attention among hydrologists. Noteworthy are flows involving several fluid

phases with interphase mass transfer, as commonly occurs when nonaqueous liquid contaminants percolate through partially saturated soils. Heterogeneity plays no less important a role in these flows. In fact, heterogeneity can exacerbate several types of instability that arise from the nonlinearity of the equations that govern these more complicated flows. The physics here are by no means well understood; we refer to Schwille (1984) for an overview.

3. HETEROGENEITY IN, GARBAGE OUT.

Having established how heterogeneity enters into groundwater models, we now examine how it leads to poor performance in standard numerical models of groundwater flow. For the remainder of this section we assume, for the sake of argument, that the modeler has "perfect" knowledge of an aquifer's heterogeneities. By this, we mean that the modeler knows the values of $K(x,y,z)$ and $S_s(x,y,z)$ at every point (x,y,z) in the aquifer. Notice that such knowledge does not imply any detailed knowledge about the *microscopic* heterogeneities associated with the tortuous interstices of the rock. We consider the case when significant spatial variations in K occur on a scale that is small compared with the size of the domain to be modeled, and for simplicity we neglect spatial variations in S_s.

The small-scale structure of K forces the modeler to use a fine discretization of the spatial domain. For example, if one uses finite differences to approximate the flow equations, then the maximum dimension h of the grid cells must be small enough to resolve the significant fluctuations in hydraulic conductivity. Finite-difference and finite-element schemes for solving the flow equation (3) yield large matrix equations to be solved for nodal values of head H at each time level in the model. Thus, smaller values of the grid mesh h lead to larger numbers of nodal heads and hence to larger and computationally more expensive matrix equations.

What is worse, smaller values of h yield more *poorly conditioned* matrices. For typical discretizations having spatial error $O(h^2)$, for example, the condition number of the matrix at each time level is $O(h^{-2})$ (see Johnson, 1987, Section 7.7). If one uses direct solution techniques such as the Cholesky decomposition, this large condition number can lead to enormous roundoff errors in the matrix solution, yielding unacceptably inaccurate values of head H. Numerically differentiating these heads to compute transport velocities via Equation (2) compounds the inaccuracies, and the result can be a useless velocity field v computed from a "perfectly" known hydraulic conductivity.

One can ameliorate the accumulation of roundoff by using iterative techniques, such as variants of relaxation schemes or conjugate gradients. Here

again, large condition numbers lead to difficulties, this time in the form of slow iterative convergence. One attractive property of conjugate-gradient techniques is that they are readily amenable to preconditioning, which can reduce this effect. Research into preconditioners that eliminate the slow convergence associated with fine discretizations is an active field of research; see Golub and O'Leary (1989) for a review.

Still, fine grids do not tell the entire story. When K varies spatially, there is a contribution to poor conditioning attributable simply to the discrepancy between the largest and smallest values, say $K_{\max}$ and $K_{\min}$, occurring in the model's domain. In fact, when one uses an iterative scheme to solve the matrix equations, the factor by which each iteration reduces the error in the approximate solution typically has the form $1 - O(K_{\min}/K_{\max})$ (see, for example, Allen et al., in preparation). Thus the convergence rate can be close to 1, and therefore prohibitively slow, when $K_{\min}$ differs from $K_{\max}$ by several orders of magnitude, independent of the grid mesh h.

Research by many investigators indicates that there is hope for this problem. Our own work, for example, suggests that a profitable first step is to isolate the two sources of poor conditioning by solving Equations (1) and (2) as a coupled system, using mixed finite-element methods (see, for example, Allen et al., 1985). The effects of highly variable conductivity K then influence only the discrete analog of Equation (2), which one can attack using any of several preconditioning schemes that effectively adapt to the heterogeneity. One can then address the effects of small grid mesh h by developing appropriate preconditioners for the conjugate-gradient method, as in Ewing et al. (to appear), or by exploiting multigrid techniques, as in Allen et al. (in preparation).

As an illustration of the potential for success in this area, we present iterative convergence rates for two schemes applied to steady-state flows in a set of fictitious aquifers (Allen et al., in preparation). The functional forms used for K in these experiments, listed in Table 1, are clearly contrived, yet in their spatial variability they can be just as troublesome as many occurring in nature. Figure 1 shows plots of iterative convergence rate versus grid mesh for each realization of K, using a scheme whose convergence rate is theoretically independent of h owing to a peculiar splitting of the mixed-method equations. This scheme overcomes sensitivity to small h but remains sensitive to spatial variations in K, as the slow convergence rates for the realization K_V attest. Table 2 displays iterative convergence rates for a modified version of the splitting scheme. Here, we precondition the discrete Darcy equations arising from the mixed method to mitigate sensitivity to

heterogeneity. Theory estimates a convergence rate of 0.5, independent of h and K, an estimate that the computed rates confirm. The methods used to generate these results by no means offer a final solution to the poor conditioning arising from heterogeneity, but they demonstrate that heterogeneity actually heightens the need for advances in numerical analysis.

The proper use of mixed finite-element methods offers the further advantage of avoiding the deterioration in accuracy that occurs when one numerically differentiates heads to compute Darcy velocities. As Ewing and Wheeler (1983) explain, mixed methods can generate approximate heads and velocities that have equal-order accuracy, a property that is especially attractive in the context of contaminant transport modeling.

Still, problems remain. We have not considered the effects of spatial variability in specific storage. Moreover, considerable work remains to be done to make mixed finite-element models truly efficient and flexible. Issues such as adaptive local grid refinement, exploitation of parallel computing architectures, and the treatment of nonlinearities associated with multiphase flows are prime examples of ongoing work along these lines.

4. STATISTICS IN, GARBAGE OUT.

In reality, heterogenities will never be "perfectly" known. The best we are likely to achieve are fairly detailed statistical descriptions of heterogeneities below some scale, which is likely to be large in practice. Thus it may suffice to use models to generate suites of scenarios yielding statistical predictions of aquifer behavior. For this strategy to be successful, ensembles of "statistically equivalent" realizations of a given heterogeneous aquifer must yield predictions that are "statistically similar." In other words, the model's predictions should be "stable," in some sense, against changes in heterogeneous structure that preserve the detailed statistics of the aquifer. Otherwise, a given statistical description of an aquifer might be consistent with a large and wildly varying class of model predictions. The notions of statistical equivalence, statistical similarity, and stability lack rigorous definition at this point, but we contend that there is considerable work to be done to make this strategy practical.

As evidence for our contention, we consider a set of numerical experiments involving coupled underground flow and solute transport. The coupling consists of the usual dependence of the solute transport coefficients on the output of a flow model, together with a dependence of the dynamic viscosity μ on the concentration c predicted by the transport model. This

model has its origin in oilfield applications, where injection of miscible fluids less viscous than oil is a common form of enhanced oil recovery. Nevertheless, there are clear analogies to be drawn with the physics of groundwater contamination and remediation. The experiments suggest that statistically similar heterogeneous porous media can yield flow fields that are qualitatively dissimilar in significant respects.

In the cases modeled, μ is a decreasing function of c, so the injected fluid is more mobile than the displaced fluid. This adverse mobility ratio makes the displacement unstable: Small perturbations in the geometry of the displacement front can lead to large differences in the fraction of the pore space contacted by the injectant. One manifestation of the instability is the occurrence of viscous fingers in the displacement front. It is not clear physically how small the perturbations can be and still trigger these fingers, but it is conceivable that they could arise from heterogeneities at the microscopic scale as well as the macroscopic scale. In this case, no model based on Equations (1) through (4) can possibly resolve all of the instabilities occurring in macroscopic flows. With this caveat in mind, we examine the effects of fine-scale but macroscopically resolvable heterogeneities on miscible displacement.

Figure 2 shows concentration isopleths for simulated displacements in two random porous media, with fluid being injected in the lower left corner and produced at the upper right (Ewing et al., 1989). The two media are independent realizations of the same lognormal spatial permeability distribution, and they have the same correlation length. The predicted displacement patterns show that the flows in the two model media yield qualitatively different concentration fields. In fact, the flows differ significantly even with respect to relatively coarse measures, such as the pore volumes of resident fluid produced after one pore volume of injection. The result for Figure 2a is 0.6921, while that for Figure 2b is 0.4968 – a decrease of over 28 percent. Clearly, these statistically similar media have dissimilar flow characteristics, at least for the physics modeled here.

Overcoming the difficulties associated with statistical characterizations of heterogeneity will require new modeling techniques and perhaps to wholly new ways of using models. Among the promising avenues for the near term are methods for scaling up fine-scale information to produce realistic models using coarse, computationally affordable grid cells. Homogenization theory (Bourgeat, 1984), flux-based averaging (White and Horne, 1987), and effective macroscopic dispersion tensors (Ewing et al., 1989) are three such approaches.

In the long run, the issue of uncertainty arising from aquifer heterogeneity has a direct bearing on the uses of deterministic models, since some uncertainty will doubtless remain as numerical analysis progresses. The inherently statistical nature of the problem reflects, in part, the discrepancy between the scales at which Equations (1) through (4) apply and the scales at which aquifers are accessible to measurement. This discrepancy implies a need for broader research into the relationships between fundamental physics, model formulation, numerical analysis, and parameter identification in groundwater modeling.

5. CONCLUSIONS.

The adages, "heterogeneity in, garbage out," and "statistics in, garbage out," are probably too pessimistic. We really intend the first adage as a caution: Modelers should not assume that all is settled on the numerical front, and that all we need are better measurements to feed into existing models. Numerical methods that are standard engineering practice today will become increasingly inadequate as better measurements of heterogeneous aquifer parameters become available. The second adage is a caution of a different sort. It suggests that the problems associated with uncertainty in heterogeneous aquifers may not be amenable to solution via straighforward discretization of the standard governing equations. Instead, these problems may require new approaches, in which rigorous numerical work contributes to the development of model formulations appropriate to the scales at which the models will actually be run.

6. ACKNOWLEDGMENTS.

The Wyoming Water Research Center supported this work. We also received support from NSF grant RII-8610680 and ONR contract N00014-88-K-0370.

7. REFERENCES.

Allen, M.B., Ewing, R.E., and Koebbe, J.V., "Mixed finite-element methods for computing groundwater velocities," *Numer. Meth. P.D.E. 3* (1985), 195-207.

Allen, M.B., Ewing, R.E., and Lu, P., "Well conditioned iterative schemes for mixed finite-element models of porous-media flows," (in preparation).

Bourgeat, A., "Homogenized behavior of two-phase flows in naturally fractured reservoirs with uniform fractures distribution," *Comp. Meth. Appl. Math. Engrg. 47* (1984), 205-216.

Ewing, R.E., Lazarov, R.D., Lu, P., and Vassilevski, P.S., "Preconditioning indefinite systems arising from mixed finite-element discretizations of second-order elliptic systems," *Proceedings, Conference on Preconditioned Conjugate Gradient Methods,* Nijmegen, Netherlands (1989), (to appear).

Ewing, R.E., Russell, T.F., and Young, L.C., "An anisotropic coarse-grid dispersion model of viscous fingering in five-spot miscible displacement that matches experiments and fine-grid simulations," *Proceedings, Eleventh SPE Symposium on Reservoir Simulation,* Houston, TX (1989), 447-465.

Ewing, R.E., and Wheeler, M.F., "Computational aspects of mixed finite element methods," in *Numerical Methods for Scientific Computing,* ed. by R.S. Stepleman, Amsterdam: North Holland (1983), 163-172.

Fried, J.J., *Groundwater Pollution: Theory, Methodology, Modelling and Practical Rules,* Amsterdam: Elsevier, 1975.

Golub, G.H., and O'Leary, D., "Some history of the conjugate gradient and Lanczos methods," *SIAM Review 31:1* (1989), 50-102.

Huyakorn, P.S., and Pinder, G.F., *Computational Methods in Subsurface Flow,* New York: Academic Press, 1983.

Johnson, C., *Numerical Solutions of Partial Differential Equations by the Finite Element Method,* Cambridge, U.K.: Cambridge University Press, 1987.

Schwille, F. "Migration of organic fluids immiscible with water in the unsaturated zone," in *Pollutants in Porous Media,* ed. by B. Yaron et al., Berlin: Springer-Verlag, 1984.

White, C.D., and Horne, R.N., "Computing absolute transmissibility in the presence of fine-scale heterogeneity," *Proceedings, Ninth SPE Conference on Reservoir Simulation,* Dallas, TX (1987), 265-278.

TABLE 1. CONDUCTIVITY FIELDS USED IN NUMERICAL EXPERIMENTS FOR ITERATIVE SOLUTIONS OF MIXED FINITE-ELEMENT MODELS FOR STEADY-STATE GROUNDWATER FLOW.

$$
\begin{aligned}
K_{\mathrm{I}}(x,y) &= 1; \\
K_{\mathrm{II}}(x,y) &= e^{-x-y}; \\
K_{\mathrm{III}}(x,y) &= \begin{cases} 1, & \text{if } x < y, \\ 0.1, & \text{if } x \geq y; \end{cases} \\
K_{\mathrm{IV}}(x,y) &= K_{\mathrm{II}}(x,y) \cdot K_{\mathrm{III}}(x,y); \\
K_{\mathrm{V}}(x,y) &= \begin{cases} 1, & \text{if } x < y, \\ 0.01, & \text{if } x \geq y. \end{cases}
\end{aligned}
$$

TABLE 2. CONVERGENCE RATES FOR A UNIFORMLY CONDITIONED ITERATIVE SCHEME APPLIED TO PROBLEMS IDENTIFIED IN TABLE 1.

	GRID MESH h				
COEFFICIENT	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
K_{I}	0.4933	0.4988	0.4993	0.4995	0.4999
K_{II}	0.4966	0.4995	0.4988	0.4997	0.4999
K_{III}	0.4948	0.4982	0.4991	0.4998	0.4999
K_{IV}	0.4947	0.4980	0.4992	0.4998	0.4999
K_{V}	0.4939	0.4978	0.4989	0.4999	0.5000

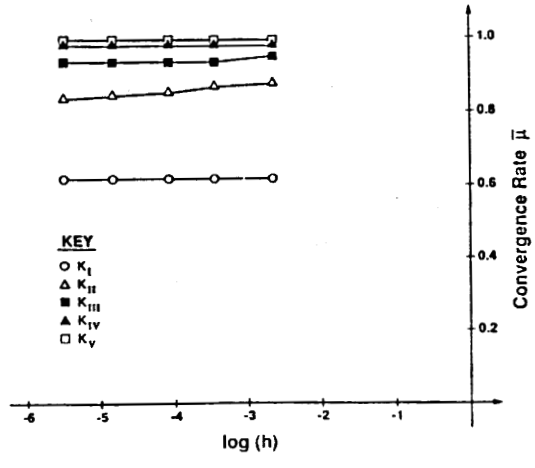
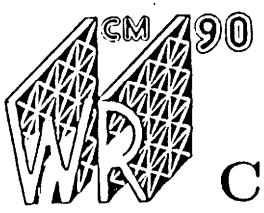


Figure 1. Convergence rates for an h-independent iterative scheme applied to problems identified in Table 1.

Figure 2. Concentration isopleths after injection of one pore volume for model miscible displacements in two random media having similar statistics.



Computational Methods in Surface Hydrology

Proceedings of the Eighth International Conference on Computational Methods in Water Resources, held in Venice, Italy, June 11-15 1990.

Editors: G. Gambolati
A. Rinaldo
C.A. Brebbia
W.G. Gray
G.F. Pinder

Computational Mechanics Publications, Southampton Boston

Co-published with

Springer-Verlag, Berlin Heidelberg New York London Paris Tokyo

An Eulerian-Lagrangian Method for Finite-Element Collocation using the Modified Method of Characteristics

M.B. Allen, A. Khosravani
Department of Mathematics, University of Wyoming, Laramie, Wyoming 82071, USA

ABSTRACT

We present a collocation method for the two-dimensional advection-diffusion equation when advection is dominant. The method uses a modified method of characteristics in conjunction with an alternating-direction algorithm to yield accurate, efficient numerical solutions.

INTRODUCTION

We discuss a collocation-based scheme for the advection-diffusion equation in two space dimensions. The scheme employs two devices to enhance its effectiveness. The first device is an alternating-direction procedure (Celia [1]) that yields highly parallelizable time-stepping algorithms. The second device is a modified method of characteristics (see Russell [2]) that improves the time-stepping error in advection-dominated flows.

The advection-diffusion equation for a steady, incompressible velocity field $\mathbf{v}(\mathbf{x})$ in two space dimensions is

$$\partial_t u + \mathbf{v} \cdot \nabla u - \nabla \cdot (D \nabla u) = 0. \qquad (1)$$

Here, $u(\mathbf{x}, t)$ is the unknown solute concentration. In porous-media applications, D accounts for the effects of hydrodynamic dispersion, which has a tensor form whose components depend on $\mathbf{v}$. In this paper, however, we take $D \geq 0$ to be a scalar constant for simplicity. We also assume that advection dominates the solute transport, in the sense that, if L is the diameter of the spatial domain, then the Peclet number $\|\mathbf{v}\|_\infty L/D \gg 1$. In this regime, it is useful to rewrite Equation (1) as follows:

$$D_t u - \nabla \cdot (D \nabla u) = 0\,. \qquad (2)$$

Here, $D_t = \partial_t + \mathbf{v} \cdot \nabla$ denotes the material derivative of the fluid-solute mixture.

We consider the following initial-boundary-value problem on the spatial domain $\Omega = (-1,1) \times (-1,1)$:

$$D_t u - \nabla \cdot (D \nabla u) = 0, \quad (\mathbf{x}, t) \in \Omega \times (0, \infty),$$

$$u(\mathbf{x}, 0) = u_I(\mathbf{x}), \qquad \mathbf{x} \in \Omega,$$

$$u(\mathbf{x}, t) = 0, \qquad (\mathbf{x}, t) \in \partial\Omega \times (0, \infty).$$

This problem is a simple model of the movement of an initial contaminant plume, so long as the plume does not approach $\partial\Omega$.

NUMERICAL METHODS

To discretize this problem in space, we use finite-element collocation on Hermite bicubics. We merely summarize this method here, referring readers to Curran and Allen [3] for more details. Let Δ be a rectangular grid on Ω, partitioning Ω into a collection of rectangular elements Ω_i bounded by adjacent grid lines $x = x_i$ and $y = y_j$. Call the mesh of this grid h. Denote by $\mathcal{M}$ the trial space of all Hermite piecewise bicubics that vanish on $\partial\Omega$. (The Hermite piecewise bicubics are functions in $C^1(\Omega)$ whose restrictions to any Ω_i are products of cubics in x with cubics in y.) Any function $\hat{u} \in \mathcal{M}$ has the form

$$\hat{u} = \sum_{i,j} \left(u_{ij} H_{00ij} + u_{ij}^{(x)} H_{10ij} + u_{ij}^{(y)} H_{01ij} + u_{ij}^{(x,y)} H_{11ij} \right),$$

where the functions $H_{pqij}(x, y)$ form a nodal basis for $\mathcal{M}$ (Prenter [4]).

To determine the nodal unknown coefficients in this expansion, we substitute $\hat{u}$ into the left side of Equation (2) and force the residual to vanish at a set of collocation points $\bar{\mathbf{x}}_m$, which for optimal-order accuracy we choose to be the 2×2 Gauss quadrature abscissae in each element Ω_i. This procedure yields precisely enough ordinary differential equations in time, each having the form

$$D_t \hat{u}(\bar{\mathbf{x}}_m, t) - \nabla \cdot [D \nabla \hat{u}(\bar{\mathbf{x}}_m, t)] = 0, \tag{3}$$

to determine the evolution of the unknown coefficents of $\hat{u}$, assuming we can project the initial function u_I onto $\mathcal{M}$ to get reasonable initial data $\hat{u}(\bar{\mathbf{x}}_m, 0)$.

We discretize Equation (3) in two steps. First, following Russell [2], we rewrite $D_t \hat{u}$ using the modified method of characteristics (MMOC). In the context of collocation, MMOC leads to a difference expression of the form

$$D_t \hat{u}(\bar{\mathbf{x}}_m) \simeq k^{-1} \left[\hat{u}^{n+1}(\bar{\mathbf{x}}_m) - \hat{u}^n(\mathbf{x}_m^*) \right],$$

where $\hat{u}^n(\mathbf{x})$ denotes an approximate value of $\hat{u}(\mathbf{x}, nk)$, k being the time step. Here, $\mathbf{x}_m^*$ is a "backtrack" point, computed according to the method of characteristics for the purely advective version of Equation (2). Specifically, if $(s(t), t)$ is a parametrization of the characteristic curve $d\mathbf{x}/dt = \mathbf{v}$ passing through $\bar{\mathbf{x}}_m$,

$$\mathbf{x}_m^* = \bar{\mathbf{x}} + \int_{t_{n+1}}^{t_n} \mathbf{v}(s(t), t)\, dt. \tag{4}$$

In practice we compute $\mathbf{x}_m^*$ approximately, as discussed below.

The second step in discretizing Equation (3) is to use an alternating-direction collocation (ADC) approach similar to that of Celia [1]. We perturb the discrete operator equations to effect the following factoring along the x- and y-coordinate directions:

$$(1 + k\mathcal{L}_x)(1 + k\mathcal{L}_y)\hat{u}^{n+1}(\bar{\mathbf{x}}_m) = \hat{u}^n(\mathbf{x}_m^*) + O(k^2). \tag{5}$$

Here, $\mathcal{L}_x = -\partial_x(D\partial_x)$ and $\mathcal{L}_y = -\partial_y(D\partial_y)$. By properly numbering the collocation equations and unknowns, one can reduce the equations (5) to an algebraic system that involves highly parallel sets of matrix equations, each of which has an inexpensive, one-dimensional structure. Curran and Allen [3] discuss efficient algorithms for solving such systems on parallel-architecture computers with shared memory. As that paper demonstrates, speedup curves of slope greater than 0.8 are attainable on an Alliant FX/8 eight-processor machine.

COMPUTATIONAL RESULTS

To illustrate the effectiveness of the scheme, we show results of a rotating plume problem in which $\mathbf{v}(x, y) = 2\pi(-y, x)$ and $D = 0$. The initial condition is a "Gauss hill" with center $(0, -0.6)$, unit height, and standard deviation $\sigma = 0.066$. We cut off the initial condition spatially, setting $u(\mathbf{x}, 0) = 0$ near $\partial\Omega$ for consistency with boundary conditions.

In this purely advective problem, $u(\mathbf{x}, 1) = u(\mathbf{x}, 0)$. We examine how well the numerical solution matches this property as we vary the time step k and the scheme used to compute the backtrack points $\mathbf{x}_m^*$. We use two backtracking schemes. The first uses an approximation $-k\mathbf{v}(\bar{\mathbf{x}}_m)$ to the integral in Equation (4). The second uses the approximation that is quadratic in k, namely, $-k\mathbf{v}(\bar{\mathbf{x}}_m) - k^2 \nabla \mathbf{v}(\bar{\mathbf{x}}_m) \cdot \mathbf{v}(\bar{\mathbf{x}}_m)/2$.

Figure 1 displays the $\hat{u} = 0.2$ contour for numerical solutions at $t = 1$, together with the true center $(x, y) = (0, -0.6)$ of the plume at $t = 1$. Shown are the contours for "linear" backtracking with $k = 0.01$ and $k = 0.004$ and for "quadratic" backtracking with $k = 0.01$. The plot suggests that, given the overall $O(k)$ accuracy of MMOC timestepping, there is only a slight gain in accuracy with the higher-order backtracking.

DISCUSSION

Several features of the ADC-MMOC approach make it an attractive one. First, the method inherits the high-order spatial accuracy associated with the finite element collocation. Percell and Wheeler [5] show that the scheme has $O(h^4)$ spatial accuracy for elliptic spatial operators. ADC attains this accuracy with "one-dimensional" matrices having bandwidth

five. A related Galerkin-based scheme using piecewise bilinear elements, described in Krishnamachari et al. [6], yields $O(h^2)$ accuracy with one-dimensional matrices having bandwidth three.

Second, the use of MMOC has additional advantages in reducing the temporal truncation error and in reducing the number of degrees of freedom needed to resolve sharp fronts. Russell [2] discusses these advantages. Another aspect of MMOC is that it effectively removes the dominant advective term from the spatial operator, leaving only the diffusive operator to be discretized via collocation. This fact is intuitively appealing, since we expect collocation on Hermite cubics to yield $O(h^4)$ accuracy for Equation (1) when $\mathbf{v} = 0$ but only $O(h^3)$ accuracy when $D = 0$ (see Dupont [7]). MMOC thus allows the collocation procedure to discretize just the spatial operator $-\nabla \cdot (D\nabla)$ for which it is best suited, even when the other spatial operator $\mathbf{v} \cdot \nabla$ is physically dominant.

Third, the ADC algorithm renders the scheme amenable to parallel processing. An interesting facet of the application of MMOC here is that it may help reduce the temporal error introduced by the spatial splitting when advection dominates. Observe that, with MMOC, the splitting in Equation (5) requires a perturbation of the form $k^2 \mathcal{L}_x \mathcal{L}_y \hat{u}$, where $\mathcal{L}_x = -\partial_x(D\partial_x)$ and $\mathcal{L}_y = -\partial_y(D\partial_y)$ are operators whose effects on $\hat{u}$ are "small". By contrast, without MMOC the ADC splitting involves the operators $\mathcal{L}_x = v_x\partial_x - \partial_x(D\partial_x)$ and $\mathcal{L}_y = v_y\partial_y - \partial_y(D\partial_y)$, in which the dominant advective terms appear. Thus we expect the splitting error to be smaller in magnitude in the MMOC version than in the original version of ADC.

These observations suggest that the ADC-MMOC approach can be a highly efficient and accurate technique for advection-dominated transport problems. There remain several avenues for further work on the method. Among these are the treatment of tensor dispersion and the incorporation of variations in the third spatial dimension.

ACKNOWLEDGMENTS

The Wyoming Water Research Center provided support for this project. We also received support from NSF grant RII-8610680 and ONR grant 0014-88-K-0370.

REFERENCES

1. Celia, M.A. Collocation on Deformed Finite Elements and Alternating Direction Collocation Methods, Ph.D. dissertation, Princeton University, 1983.

2. Russell, T.F. An Incompletely Iterated Characteristic Finite Element Method for a Miscible Displacement Problem, Ph.D. dissertation, University of Chicago, 1980.

3. Curran, M.C., and Allen, M.B., Parallel Computing for Solute Transport Models via Alternating-Direction Collocation, Advances in Water Resources, to appear.

4. Prenter, P.M. Splines and Variational Methods, Wiley, New York, 1975.

5. Percell, P., and Wheeler, M.F., A C^1 Finite Element Collocation Method for Elliptic Equations, SIAM J. Numer. Anal. Vol. 17, No. 5, pp. 605-622, 1980.

6. Krishnamachari, S.V., Hayes, L.J., and Russell, T.F., A Finite Element Alternating-Direction Method Combined With a Modified Method of Characteristics for Convection-Diffusion Problems, SIAM J. Numer. Anal., to appear.

7. Dupont, T., Galerkin Methods for First-Order Hyperbolics: An Example, SIAM J. Numer. Anal. Vol. 10, pp. 890-899, 1973.

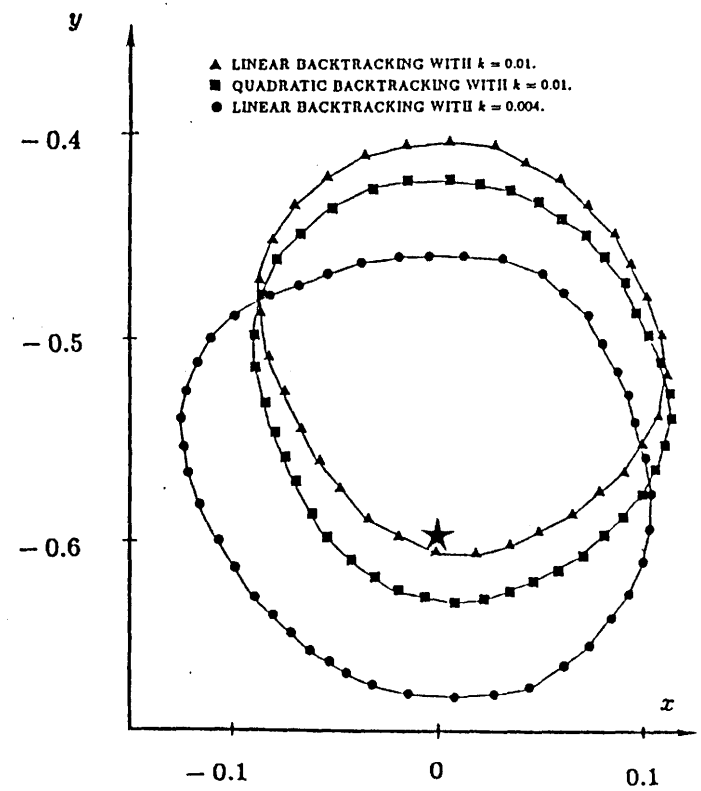


Figure 1. Comparison of $\hat{u}(x, 1) = 0.2$ contours for the ADC-MMOC method applied to an advective rotating plume problem. Shown are contours for timestep $k = 0.01$ with linear and quadratic backtracking and for timestep $k = 0.004$ with linear backtracking. The symbol $\star$ marks the center of the plume in the exact solution $u(x, 1)$.

USER'S GUIDE TO ADMOC: A CONTAMINANT TRANSPORT CODE BASED ON ALTERNATING-DIRECTION COLLOCATION AND A MODIFIED METHOD OF CHARACTERISTICS

by
Azar Khosravani
Myron Allen
Department of Mathematics
University of Wyoming
Laramie, WY 82071

August, 1990

Note: ADMOC is a research code developed at the University of Wyoming. It is not intended as a ready-to-use groundwater transport model, nor is it suitable as public-use software. The authors make no claims for applicability of the code to problems more general than the sample problems discussed in this document. In particular there is no agreement, explicit or implied, that the authors will provide assistance to users wishing to install, debug, or use the code.

INTRODUCTION

ADMOC is a Fortran code that employes a modified method of characteristics combined with an alternating-direction algorithm to solve a two-dimensional advection-diffusion equation. For more detail on the mathematical formulation of the code, refer to Allen and Khosravani [1].

The purpose of this document is to show first-time users how to use the code. We assume the user works on a local area network like that available in the University of Wyoming Mathematics Department. The main computing machines are Unix-based: an Alliant FX/8 for numerical processing and a Silicon Graphics Iris for graphic processing. We use an ATT workstation running MS/DOS for transferring codes and data from floppy disk.

The document starts by explaining how to transfer the files from the enclosed floppy disk into your Alliant account. Then it gives and explains all the commands that need to be employed to run the code. The paper also demonstrates in detail how to view the output files by using the graphic software available on the Iris. Fianlly, it explains how to get printouts of the files.

TRANSFERRING FILES TO THE ALLIANT

First you need to find a machine with a $5\frac{1}{4}$ inch floppy disk drive and hard drive, with capability to transfer data to the Alliant using the utility FTP. The Math Department's ATT workstation, Ahab, with floppy drive B and hard disk C, can be used for this file transfer. Throughout this paper \$ is used to denote the machine prompt. Explanations of commands appear in parentheses. User responses appear in boldface. While logged in to Ahab, with the floppy disk in drive B, proceed as follows:

$ftp mercury

('mercury' is the network name for Alliant.)

$username:**myname**

$password:**mypass**

user logged in on mercury.

$lcd B:

(local directory change on ATT to B drive, which is the floppy disk drive.)

$mputB:*.*

(In response to this command, the machine will ask you, file by file, whether you want the file copied to the Alliant. Answer 'y', followed by a RETURN, to each question. In all, 18 files should be transferred from the floppy disk by this command.)

$put makefile

$put 1fig

$put 5fig

$lcd C:

(local directory change to C drive)

$bye

(takes you out of the FTP utility and back to Ahab's operating system, MS/DOS.)

Now you have copies of all the files on the floppy disk in your Alliant account, but these files reside in the top directory. To keep your Alliant

account manageable, make a directory, say 'adccode', and transfer all these files to that directory. The following commands will accomplish this task from Ahab or most other terminals linked to the University of Wyoming campus network.

$telnet junior

(establishes a connection to the Iris. You cannot log into the Alliant directly from Ahab.)

$username:**myname**

$password:**mypass**

$telnet mercury

(establishes a connection to the Alliant.)

$username:**myname**

$passoword:**mypass**

(If your *.login* file on the Alliant doesn't automatically set your terminal type to vt100, type

$set term=vt100

in response to the next prompt. If you're not sure what a *.login* file is or don't know if you have one, type the above command anyway.)

$mkdir adccode

(makes a subdirectory named 'adccode')

$cp*.* adccode

$cp makefile adccode

$cp 1fig adccode

$cp 5fig adccode

(copies every file to the directory adccode. Now, you want to go up to the top directory and erase those files from it.)

$cd. .

(takes you to the directory above where you are sitting.)

$rm *.f

(removes all your fortran files in the top directory.)

$rm *.in

(removes all your *.in* files in the top directory.)

$rm makefile

$rm 1fig

$rm 5fig

This procedure assumes that you have a fresh account on the Alliant. Otherwise, you should have first made the subdirectory 'adccode' and then transferred files from the floppy disk to that subdirectory.

RUNNING THE CODE

To make a successful run of the ADMOC, the user needs to go through the following steps:

a) Getting your program ready for a run. That is, adjusting the data file, the parameters, and the velocity field to the problem you want to solve.

b) Running the code and thus generating the graphic output files.

c) Viewing the graphic files to examine the correctness of the results.

Now we discuss these steps. Refer to the flowchart at the end of this user's guide for information about the subroutine structure of the code.

a) Getting your program ready for a run.

The input data are read in the main program, *adc.f*, from the data file *adc.in*. The program currently assumes that the initial contaminant plume is a 'Gauss hill' (i.e., binormal distribution in space) with specified center of mass and standard deviation. The input data include:

DT - size of time-steps

THETA - timestepping parameter (0 gives an explicit scheme, and 1 gives an implicit scheme.)

XDIM and YDIM - length of the domain in the x- and y-directions, respectively.

X1 and Y1 - the starting point in x- and y-directions, respectively, for the spatial domain.

SIGMA - standard deviation of the initial plume.

X0 and Y0 - the origin of the concentration plume.

XNODE and YNODE - numer of nodes in x- and y-directions.

You should check and possibly adjust the parameters MNON, MNOCP, NROW are used in the subroutines *adc.f, cchar.f, init.f, kindex.f, print.f, reformz.f,* and *update.f.* MNON respresents the maximum number of nodes and is obtained by multiplying XNODE by YNODE. MNOCP is the maximum number of collocation points; its value is 4 times the number of elements in the domain. To be more precise, MNOCP = 4(XNODE-1)×(YNODE-1). NROW is the number of collocation points in one row in the x-direction, that is, NROW=2(XNODE-1). If the parameter values in the Fortran code are not consistent with

the data in *adc.in*, you must edit the Fortran code to change the parameter statements.

Suppose we want to make three different runs using the three different sample data files provided in this directory. In our first run we use the file *adc1.in*, which gives the rotating velocity field $\mathbf{v} = 2\pi(-y, x)$. The spatial domain for this run is $(-1, 1) \times (-1, 1)$, and 101 nodes are used in each coordinate direction. The initial plume is centered at the point (x, y)=(0,-0.5), and it has standard deviation 0.16; it completes one rotation in 100 time steps. Every tenth time step is printed since NPRINT=10.

The velocity field is defined in the subroutine *cchar.f*. The velocity fields 'rotating plume' and 'shear' are made available to the user in the subroutine. To activate the desired velocity field, just 'comment out' all the other velocity fields that are provided (by placing a 'c' in column 1 of each line of the source code) and uncomment the wanted velocity field. You may also define a new velocity field by writing appropriate Fortran code in the subroutine *cchar.f*.

Normally, we would need to go through steps b) and c), described below, to see the results of this run. Before describing these steps, though, we describe the remaining two sample runs.

In our second run, we use the same spatial domain as the first run. We center the initial plume at a different location and use the shear-flow velocity field $vx = \frac{1}{2}(1 + y)$ and $vy = 0$. To make these changes, we use the input file *adc2.in*. Now we need to edit the files *adc.f* and *makefile*, changing every occurrence of *adc1.in* to *adc2.in*. We also need to comment out the rotating velocity field and uncomment the shear-flow velocity field in *cchar.f*.

Our third run is quite different from the first two runs. Here we use a rectangular domain and leave it up to the user to choose the velocity field. It can be the rotating field, the shear field, or a new velocity field defined by the user in *cchar.in*. Here we use the data file *adc3.f* and so we need to change *adc.f* and *makefile* just as we did when we used *adc2.in*. Notice that we need to use a different graphic file here, one that accomodates the rectangular domain.

b) Compiling and executing

To compile the program on the Alliant just type **make**. Minor modifications of the *makefile* routine might be necessary to compile the program on another machine. The command **make** causes the Alliant to produce an executable file called *adc.x*. To run the program just type the name of the executable file, *adc.x*. Since this program takes up to an hour to run, it is advisable to run it in background, that is, to assign it a low priority on the machine's scheduler to avoid interfering with other users. This is what you need to do:

$cd adccode

(changes directory to adccode)

$make

(compiles program adc)

$adc.x &

(The command 'adc.x' causes the program *adc* to run, and the modifier '&' puts this job in background.)

Running the *adc.x* produces data files such as *adc00000*,..., *adc00100* to be read by the graphic file *visions*. One way to check whether the run is completed is to list the files in your directory. If *adc00100*, which

is the last output file that the program generates, is in your directory, then the run is over. To get a list of files in the directory, type

$ls

If the run is not over and you want to know how much time the machine allocated to running your program, type:

$ps -aux

(This command lists all the jobs presently running on the machine along with some information about them, such as how much computer time has been allocated to them.)

c) Viewing the graphics

At the present we are running our program on the Alliant FX/8 and using the Iris to view the output files. This means that we need to take our output files from the Alliant to the Iris. To do so, we use the utility FTP. For example, suppose we are in the directory 'adccode' on the Alliant that contains our output files, namely, *adc00000,...*, *adc00100.* We want first to create directory 'mygraphics' on the Iris and then to transfer our output files from 'adccode' on the Alliant to 'mygraphics' on the Iris. While you are logged into the Alliant, proceed as follows:

$rlogin junior

('junior' is the network name for the Iris.)

$username:myname

$password:mypass

(user logged in)

$mkdir mygraphics

(makes a subdirectory named 'mygraphics')

$lo

(log out command; now you are back on the Alliant.)

$ftp junior

$username:**myname**

$password:**mypass**

$cd mygraphics

(changes directory to 'mygraphics' on the Iris.)

$mput adc00*

(copy all the Alliant files starting with *'adc00'* in the subdirectory 'mygraphics' on the Iris. The utility will ask you, file by file, whether you want the file to be transferred. Respond with 'y', followed by a return, for each file. If 'mput' does not work, then simply transfer the files one by one using the FTP command 'put'.)

$put 1fig

$put 5fig

(*1fig* and *5fig* give data to the graphic software on the Iris.)

$bye

(Now you are back to your Alliant account.)

Now log out of the Alliant, go to the Iris workstation, and log into your account on that machine. The Iris workstation is an Iris-ansi terminal, and you have to set it accordingly as follows:

$set term=iris-ansi

If you are working with a square grid, for instance, using *adc1.in* or *adc2.in*, all you have to do on the Iris is type:

$visions 1fig

While you are in the subdirectory 'mygraphics'.

A small red square will appear on the screen; this is a window that you need to open up. Using the mouse, move the arrow on the monitor to the upper left corner of the screen and, while holding the right button, move the arrow to the lower right corner of the screen. Now let go of the right button. The viewing window is now open and you are looking at a graph of the data stored in the first output file, *adc00000*, which records the initial time step. Moving the mouse arrow inside the blue domain and holding the right button down initiates the animation option, which can be activated by letting go of the right button when the arrow points to the words 'animation on'. You now are looking at animated perspective plots of the data stored in the output files.

To close the graphics window, place the arrow on the bar at the top of the screen and hold down the right mouse button. A window with different command options appears on the screen. While holding the right mouse button down, place the arrow on the 'Quit ISC Visions' option. Now let go of the right mouse button to close the graphics window.

GETTING HARD COPY OF SOURCE FILES

The Alliant directory into which you dumped the contents of the floppy disk contains the following files:

adc.f, adc1.in, adc2.in, adc3.in, cchar.f, herm.f, hermbc.f, init.f, interpolation.f, kindex.f, loadz.f, makefile.f, matrix.f, plotter.f, print.f, reformz.f, rhsx.f, solve.f, update.f.

The flowchart at the end of this document shows how these files work together. To get a hard copy of one or more of these files in Ross Hall, one needs to copy these files to the Sun workstation, indentified as 'sunrise' on the network. For a printout on the line printer, type

$print filename

While you are logged onto the Sun, to get a laser printout, type

$imprint -Psunset filename

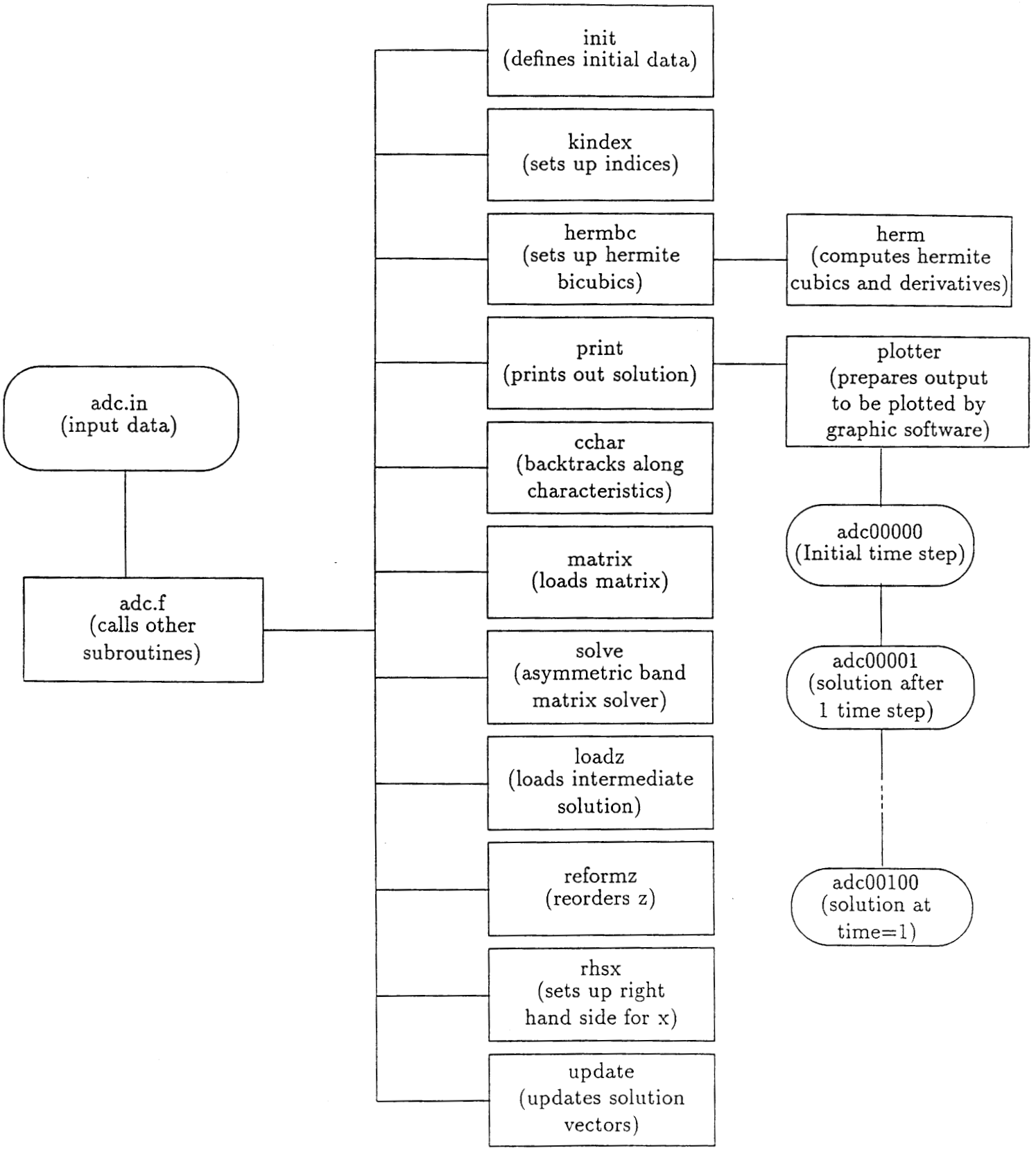
More than one file can be sent to the printers simply by listing more than one filename. For instance, type

$imprint -Psunset adc.f cchar.f herm.f

to get laser printouts of *adc.f, cchar.f and herm.f*

REFERENCE

1. Allen, M. B. and Khosravani, A., An Eulerian-Lagrangian method for finite-element collocation using the modified method of characteristics, in *Computational Methods in Surface Hydrology*, ed. by G. Gambolati et al., Berlin: Springer-Verlag, 1990, pp. 375–379.



Flowchart of the transport code ADMOC