

AD-A096 652

WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER

F/G 12/1

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY--ETC(U)

OCT 80 A BRANDT, C W CRYER

DAAG29-80-C-0041

UNCLASSIFIED

MRC-TSR-2131

NL

1 of 2
AD A
D096652



AD A 096652

MRC Technical Summary Report #2131 ✓

MULTIGRID ALGORITHMS FOR THE SOLUTION
OF LINEAR COMPLEMENTARITY PROBLEMS
ARISING FROM FREE BOUNDARY PROBLEMS

Achi Brandt and Colin W. Cryer

Mathematics Research Center
University of Wisconsin-Madison
610 Walnut Street
Madison, Wisconsin 53706



October 1980

(Received April 3, 1980)

Approved for public release
Distribution unlimited

Sponsored by

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

National Science Foundation
Washington, D.C. 20550

81 3 19 063

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR
COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS

Achi Brandt^{*, (1)} and Colin W. Cryer^{**, (2)}

Technical Summary Report #2131
October 1980

ABSTRACT

We show that the multigrid algorithms of Brandt can be adapted to solve linear complementarity problems arising from free boundary problems. The multigrid algorithms are significantly faster than previous algorithms. Using the multigrid algorithms, which are simple modifications of multigrid algorithms for equalities, it is possible to solve the difference equations to within truncation error using less work than the equivalent of six Gauss-Seidel sweeps on the finest grid.

AMS (MOS) Subject Classifications: 35J65, 35R35, 65N99, 90C33

Key Words: Multigrid Algorithms, Free Boundary Problems, Linear Complementarity Problems

Work Unit Number 3 (Numerical Analysis and Computer Science)

* The Weizmann Institute of Science, Department of Applied Mathematics, Rehovot, Israel.

** Computer Sciences Department and Mathematics Research Center, University of Wisconsin-Madison, Madison, WI 53706.

-
- (1) Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.
(2) Sponsored by the National Science Foundation under Grant No. MCS77-26732 and the United States Army under Contract No. DAAG29-80-C-0041.

SIGNIFICANCE AND EXPLANATION

Several free boundary problems, (including: saturated-unsaturated flow through porous dams; elastic-plastic torsion; and cavitating journal bearings) can be formulated as linear complementarity problems of the following type. Find a non-negative function u which satisfies prescribed boundary conditions on a given domain and which, furthermore, satisfies a linear elliptic equation at each point of the domain where u is greater than zero. We show that the multigrid algorithms of Brandt, (in which solutions are computed on a series of nested grids) which were developed to solve boundary value problems for elliptic partial differential equations, can easily be adapted to handle linear complementarity problems. The resulting algorithms are significantly faster than previous algorithms in which only one grid is used, since the computation time is proportional to the number of gridpoints on the finest grid.

Accession For

MIS CRA&I	X
MIS TAB	☐
Unrecorded	☐
Accession	

Section

A

The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the authors of this report.

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR
COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS

Achi Brandt^{*,(1)} and Colin W. Cryer^{**, (2)}

1.1 INTRODUCTION.

Several free boundary problems can be reformulated in the form of an (infinite-dimensional) LCP (linear complementarity problem): Given a polygonal domain $\Omega \subset \mathbb{R}^n$ with boundary $\partial\Omega$, and given functions f and g , find u (defined on Ω) such that (in an appropriate weak sense)

$$\begin{aligned} \text{(a)} \quad & Lu(x) \leq f(x), \quad x \in \Omega, \\ \text{(b)} \quad & u(x) \geq 0, \quad x \in \Omega, \\ \text{(c)} \quad & u(x)[Lu(x) - f(x)] = 0, \quad x \in \Omega, \\ \text{(d)} \quad & u(x) = g(x), \quad x \in \partial\Omega. \end{aligned} \tag{1.1}$$

where L is a given second order elliptic operator. The restriction that Ω is polygonal is not essential, but suffices for our present purposes. We do not write (1.1a) in the more usual form $-Lu(x) + f(x) \geq 0$ because we wish to maintain compatibility with the notation in previous papers by Brandt.

Well-known examples of free boundary problems which can be written in the form (1.1) include porous flow through dams (a recent reference is Baiocchi [1978]), journal bearing lubrication (Cryer [1971a], Cimatti [1977]) and elastic-plastic torsion (Cea, Glowinski, and Nedelec [1974], Lanchon [1974], Cryer [1979]). General references include: Duvaut and Lions [1976]; Glowinski, Lions, and Tremolieres [1976], and Cryer [1977], Glowinski [1978]; Cottle, Giannessi, and Lions [1980]; and Kinderlehrer and Stampacchia [1980].

* The Weizmann Institute of Science, Department of Applied Mathematics, Rehovot, Israel

** Computer Sciences Department and Mathematics Research Center, University of Wisconsin-Madison, Madison, WI 53706.

(1) Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

(2) Sponsored by the National Science Foundation under Grant No. MCS77-26732 and the United States Army under Contract No. DAAG29-80-C-0041.

If Ω is approximated by a regular grid then the grid can be divided into $N = |G|$ "interior" points G and $|\partial G|$ "boundary" points ∂G . Let the grid size be h . When (1.1) is approximated using finite differences on G , one obtains a (finite-dimensional) LCP:

$$\begin{aligned} (a) \quad & LU(x) \leq f(x), \quad x \in G, \\ (b) \quad & U(x) \geq 0, \quad x \in G, \\ (c) \quad & U(x)[LU(x) - f(x)] = 0, \quad x \in G, \\ (d) \quad & U(x) = g(x), \quad x \in \partial G, \end{aligned} \tag{1.2}$$

where $U(x)$ is an approximation to $u(x)$ at the grid points $x \in G \cup \partial G$ and where L is a difference operator which approximates $\mathcal{L}$. The coefficients of L are $O(h^{-2})$.

By multiplying (1.2) by h^2 and eliminating the known values of $U(x)$ on ∂G , the LCP (1.2) may be written in matrix form

$$\begin{aligned} (a) \quad & AU \leq b, \\ (b) \quad & U \geq 0, \\ (c) \quad & U^T(AU - b) = 0, \end{aligned} \tag{1.3}$$

where U is the N -vector of values of $U(x)$ on G , and A is an $N \times N$ matrix with coefficients which are $O(1)$. Since we will assume that A is symmetric and negative definite, (1.3) could be brought into the canonical form for an LCP by multiplying (1.3a) by -1 .

For example, if $\mathcal{L}$ is the Laplace operator in R^2 , then a possible choice for L would be the classical five-point difference operator, in which case A would be a matrix with diagonal elements -4 and off-diagonal elements either 0 or 1 .

The general structure of a finite-dimensional LCP is that we have a pair of vector inequalities together with the complementarity condition which states that at every point at least one of the inequalities must in fact be an equality.

There is an extensive literature on the (finite-dimensional) LCP (see Balinski and Cottle [1978]). In particular, if A is negative definite, as we assume, then there exists a unique solution to (1.2) and (1.3).

Since the LCP (1.3) arises from a free boundary problem, the matrix A has special properties which make it possible to use specialized algorithms which are particularly efficient. Such algorithms include projected SOR (Cryer [1971], Glowinski [1971]) the method of Cottle and Sacher [1977], and the modified block SOR (MBSOR) method of Cottle, Golub, and Sacher [1978]; Cryer [1979a] summarizes these algorithms and Cottle [1974] gives numerical comparisons between them.

Recently, it has been found (Brandt [1977], Brandt and Dinar [1979]) that multigrid algorithms are an effective tool for solving linear equations of the form

$$AX = b. \quad (1.4)$$

The basic idea of these multigrid algorithms is to compute on a sequence of nested grids. The computation proceeds on a particular grid until the error becomes smooth and the rate of convergence slows, at which point the computation is transferred to a coarser grid. When the error has been reduced on the coarser grid, the solution on the finer grid is corrected using interpolated values from the coarser grid.

In this paper, we show how the multigrid algorithms FAS and FMG of Brandt can be modified to solve the LCP (1.3). We find that the modified multigrid algorithms are substantially faster than previous algorithms. Indeed, with only minor modifications, the standard multigrid programs solve the LCP with essentially the same efficiency as is attained for linear equations.

The paper is organized as follows. In Section 2, we describe PFAS, the projected full approximation scheme for solving (1.3); PFAS combines the concepts of multigrid algorithms with those of projected SOR. In Section 3, we discuss the implementation of PFAS, and in Section 4, we give numerical results obtained using PFAS. In Section 5, we discuss alternative implementations of PFAS, the last of which leads to substantially improved convergence (we also include several unsuccessful implementations because they are instructive).

In Section 6, we describe results obtained using PFMG, the projected full multigrid algorithm for solving (1.3). The basic idea of PFMG is to compute the initial

approximation on each grid by interpolating an accurate solution on the next coarser grid. Using PFMG we are able to solve the LCP to within truncation error using less work than the equivalent of six Gauss-Seidel sweeps on the finest grid.

Our results are summarized in Section 7 and some possible extensions are mentioned. Finally, listings of the programs are given in the appendices.

2. PFAS (PROJECTED FULL APPROXIMATION SCHEME).

Brandt [1977], and Brandt and Dinar [1979] give a detailed exposition of multigrid methods and their philosophy, and the reader is referred to these papers for background information. The algorithm described below, PFAS, is a modification of the FAS (Full Approximation Scheme) which is considered in Section 5 of Brandt [1977], and Section 2.2 of Brandt and Dinar [1979].

The polygonal domain $\Omega \subset \mathbb{R}^n$ is approximated by a sequence of grids

$$G^1 \subset G^2 \subset \dots \subset G^M \subset \mathbb{R}^n,$$

with corresponding grid sizes

$$h_1 = 2h_2 = 4h_3 = \dots = 2^{M-1}h_M.$$

Let F^k be the restriction of f to G^k ,

$$F^k(x) = f(x), \quad x \in G^k. \quad (2.1)$$

Then, on G^k the difference equations (1.2) approximating (1.1) take the form

$$\begin{aligned} (a) \quad & L^k U^k(x) \leq F^k(x), \quad \text{in } G^k, \\ (b) \quad & U^k(x) \geq 0, \quad \text{in } G^k, \\ (c) \quad & U^k(x) [L^k U^k(x) - F^k(x)] = 0, \quad \text{in } G^k, \\ (d) \quad & U^k(x) = g(x), \quad \text{in } \partial G^k. \end{aligned} \quad (2.2)$$

Let the points of G^k be ordered: $x_1^k, x_2^k, \dots, x_{N_k}^k \in G^k$, and let U^k be the vector

$$U^k = \{U_j^k : 1 \leq j \leq N_k\} \equiv \{U^k(x_j^k) : 1 \leq j \leq N_k\}.$$

Then, (1.3) takes the form

$$\begin{aligned} (a) \quad & A^k U^k \leq b^k, \\ (b) \quad & U^k \geq 0, \\ (c) \quad & (U^k)^T [A^k U^k - b^k] = 0. \end{aligned} \quad (2.3)$$

where

$$A^k = \{a_{ij}^k : 1 \leq i, j \leq N_k\}, \quad (2.4)$$

is a known sparse symmetric negative definite matrix and $b^k = \{b_j^k\}$ is a known vector with components $b_j^k = h_k^2 F^k(x_j^k)$ (except at points x_j^k adjacent to ∂G^k).

THE PROJECTED GAUSS-SEIDEL ALGORITHM

It is possible to solve the LCP's (2.2) and (2.3) using the projected Gauss-Seidel algorithm which we now describe.

Let $u^{k,0}(x)$ be an approximate solution of (2.2) and (2.3). We compute recursively a sequence of approximations $u^{k,1}(x), u^{k,2}(x), \dots$, as follows. Let $u^{k,s-1}(x)$ be given. From (2.2d), the boundary values of $u^{k,s}(x)$ are equal to $g(x)$. The interior values of $u^{k,s}(x)$, which together comprise the vector

$$u^{k,s} = \{u_j^{k,s} : 1 \leq j \leq N_k\} = \{u_j^{k,s}(x_j^k) : 1 \leq j \leq N_k\}, \quad (2.5)$$

are obtained, point by point, by first applying the classical Gauss-Seidel method to (2.3) to obtain

$$\begin{aligned} u_j^{k,s-\frac{1}{2}} &= u_j^{k,s-1} + [b_j^k - \sum_{\ell < j} a_{j\ell}^k u_\ell^{k,s} - \sum_{\ell > j} a_{j\ell}^k u_\ell^{k,s-1}] / a_{jj}^k, \\ &= u_j^{k,s-1} + \bar{r}_j^{k,s} / a_{jj}^k, \text{ say,} \end{aligned} \quad (2.6)$$

and then projecting:

$$u_j^{k,s} = \max\{0, u_j^{k,s-\frac{1}{2}}\}. \quad (2.7)$$

The process of applying (2.6) and (2.7) for $1 \leq j \leq N_k$ to obtain $u^{k,s}$ from $u^{k,s-1}$ will be called a G^k projected Gauss-Seidel sweep, or a G^k projected sweep. The quantities $\bar{r}_j^{k,s}$ will be called the dynamic residuals.

It is known (Cryer [1971], Glowinski [1971]) that $u^{k,s} \rightarrow u^k$ as $s \rightarrow \infty$.

When implementing the projected Gauss-Seidel method only the latest values of the solution are stored. We will, therefore, often suppress the iteration counter s and denote one projected Gauss-Seidel sweep applied to (2.2) and (2.3) by

$$u^k \leftarrow \text{Projected Gauss-Seidel } [u^k : L^k, F^k]. \quad (2.8)$$

Similarly,

$$\nabla u^k = u^{k,s} - u^{k,s-1} \quad (2.9)$$

will denote the difference between the latest approximation u^k and its predecessor, while

$$v_{old}^k = u^{k,s-1} - u^{k,s-2}, \quad (2.10)$$

denotes the previous difference.

ERROR ESTIMATES FOR THE PROJECTED GAUSS-SEIDEL ALGORITHM

When implementing the projected Gauss-Seidel algorithm as part of a multigrid process, it is important to be able to estimate the error. In order to do so, we note that since, by assumption, $-A^k$ is symmetric and positive definite, there exists a coercitivity constant $\alpha_k > 0$ such that

$$w^T(-A^k)w \geq \alpha_k w^T w, \quad (2.11)$$

for all $w \in R^{N_k}$.

Lemma 2.1

Let u^k be the solution of the LCP (2.3), and let $u^k \geq 0$ be an approximate solution. Let

$$r^k = (r_j^k) = b^k - A^k u^k, \quad (2.12)$$

and $r_+^k = (r_{+j}^k)$, where

$$r_{+j}^k = \begin{cases} r_j^k, & \text{if } u_j^k > 0, \\ \min\{0, r_j^k\}, & \text{if } u_j^k = 0. \end{cases} \quad (2.13)$$

Then

$$(u^k - u^k)^T(-A^k)(u^k - u^k) \leq (u^k - u^k)^T(-r_+^k). \quad (2.14)$$

Hence,

$$\|u^k - u^k\|_2 \leq \alpha_k^{-1} \|r_+^k\|_2. \quad (2.15)$$

Proof: With r_+^k defined as above, we see that u^k satisfies the LCP:

$$\begin{aligned} (a) \quad & A^k u^k \leq b^k - r_+^k, \\ (b) \quad & u^k \geq 0, \\ (c) \quad & (u^k)^T (A^k u^k - b^k + r_+^k) = 0. \end{aligned} \quad (2.16)$$

Following Falk [1974] we multiply (2.3a) by the non-negative vector $(u^k)^T$ and use the complementarity condition (2.3c) to obtain

$$(u^k - u^k)^T A^k u^k \leq (u^k - u^k)^T b^k. \quad (*)$$

Similarly, multiplying (2.16a) by $(u^k)^T$ we obtain

$$(u^k - u^k)^T A^k u^k \leq (u^k - u^k)^T (b^k - r_+^k). \quad (**)$$

Adding (*) and (**) and combining terms we obtain (2.14) and hence (2.15). $\square$

Lemma 2.2.

Let u^k be the solution of the LCP (2.3), and let $u^k \geq 0$ be an approximate solution obtained after one or more G^k projected sweeps. Let

$$A^k = (D^k - L^k - P^k) \quad (2.17)$$

where D^k is diagonal, and L^k and P^k are strictly lower and upper triangular matrices, respectively.

Then u^k satisfies the LCP

$$A^k u^k \leq b^k - P^k \nabla u^k,$$

$$u^k \geq 0, \quad (2.18)$$

$$(u^k)^T (A^k u^k - b^k + P^k \nabla u^k) = 0.$$

Hence,

$$\|u^k - u^k\|_2 \leq \alpha_k^{-1} \|P^k\|_2 \|\nabla u^k\|_2. \quad (2.19)$$

Proof: Consider the projected Gauss-Seidel method defined by (2.6) and (2.7). For each point x_j^k we first compute the dynamic residual $\hat{r}_j^{k,s}$. The new value of $u_j^{k,s}$ is chosen so as to reduce the residual. Denote the residual at the point x_j^k immediately after step (2.7) by $\hat{r}_j^{k,s}$, so that

$$\hat{r}_j^{k,s} = \hat{r}_j^{k,s} - a_{jj}^k (u_j^{k,s} - u_j^{k,s-1}). \quad (2.20)$$

Remembering that A^k is negative definite, and hence $a_{jj}^k < 0$, we see that there are two possibilities:

$$\text{either } u_j^{k,s} > 0 \text{ and } \hat{r}_j^{k,s} = 0,$$

$$\text{or } u_j^{k,s} = 0 \text{ and } \hat{r}_j^{k,s} \geq 0.$$

Thus, dropping the superscript s , and setting $\hat{r}^k = \{\hat{r}_j^k : 1 \leq j \leq N_k\}$,

$$\begin{aligned}
u^k &\geq 0, \\
\hat{r}^k &\geq 0, \\
(u^k)^T \hat{r}^k &= 0.
\end{aligned}
\tag{2.21}$$

Let

$$r^k = b^k - A^k u^k.$$

It is readily seen from (2.17) that

$$\begin{aligned}
r^k &= \hat{r}^k + P^k (u^{k,s} - u^{k,s-1}), \\
&= \hat{r}^k + P^k \nabla u^k.
\end{aligned}
\tag{2.22}$$

Combining (2.21) and (2.22) we obtain (2.18). Comparing (2.16) and (2.18) we see that the arguments which led to (2.15) from (2.16) may be applied to (2.18), with r_+^k replaced by $-P^k \nabla u^k$, to obtain (2.19). $\square$

As Lemmas 2.1 and 2.2 show, we can estimate the error in an approximate solution u^k in terms of the residual r^k or the difference ∇u^k ; we will usually use ∇u^k to estimate the error, since this quantity is readily available during a G^k projected sweep.

Remark. The reader may wonder why we bothered to introduce r_+^k in Lemma 2.1, since (2.15) holds with r_+^k replaced by r^k . The reason is that for the LCP (2.3) there may be large positive residuals at points x_j^k where $U^k(x_j^k) = 0$, but this does not mean that the error is large. $\square$

In multigrid algorithms it is necessary to compare norms on different grids. We, therefore, wish to introduce a norm which is not grid dependent. To do so, we proceed as follows.

We first note that, to a good approximation, the coercivity constant α_k for $-A^k$ satisfies

$$\alpha_k \doteq \alpha h^2,$$

where α is the smallest eigenvalue of L .

Next, assume that the approximate grid function u^k has been extended to a function $u^k(x)$ on Ω approximating the solution $u(x)$ of (1.1). Then

$$\begin{aligned}
\|u(x) - u^k(x)\|_{2,\Omega} &= \left| \int_{\Omega} |u(x) - u^k(x)|^2 dx \right|^{\frac{1}{2}}, \\
&\approx \left| \sum_{j=1}^{N_k} h_k^n |u_j^k - u_j^k|^2 \right|^{\frac{1}{2}}, \\
&= h_k^{\frac{n}{2}} \|u^k - u^k\|_2, \\
&\leq \frac{h_k^{\frac{n}{2}}}{\alpha_k} \|p^k\|_2 \|v u^k\|_2, \\
&\approx \frac{\|p^k\|_2}{\alpha} h_k^{\frac{n}{2}-2} \|v u^k\|_2.
\end{aligned}$$

The norms $\|p^k\|_2$ are essentially independent of k ; for example, for the five-point formula, $\|p^k\|_2 \leq 2$. Thus a measure for the error $\|u(x) - u^k(x)\|_{2,\Omega}$ is provided by

$$\|v u^k\|_G \approx h_k^{\frac{n}{2}-2} \|v u^k\|_2, \quad (2.23)$$

and this norm will be used in the computations.

PFAS (PROJECTED FULL APPROXIMATION SCHEME).

PFAS (Projected Full Approximation Scheme) obtains an approximation $\bar{u}^M$ to the solution U^M on the finest grid G^M by recursively generating a sequence of approximations $\bar{u}^k$ on the grids G^k .

Each $\bar{u}^k$ is an approximate solution to an LCP of the form (2.2) with F^k replaced by a function $\bar{F}^k$ which is defined later. In general, $\bar{F}^k$ is different from F^k so that $\bar{u}^k$ is not an approximation to U^k . However, $\bar{F}^M = F^M$ and so $\bar{u}^M$ is an approximation to U^M .

We begin by initializing $\bar{u}^M$ to some suitable value. For example, we might set

$$\begin{aligned}
\bar{u}^M(x) &= g(x), \quad \text{on } \partial G^M, \\
\bar{u}^M(x) &= 0 \quad \text{in } G^M.
\end{aligned} \quad (2.24)$$

We also set

$$\|v \bar{u}^M\|_G = 10^{30}, \quad \epsilon^M = \epsilon, \quad (2.25)$$

(where ϵ is the desired accuracy on the finest grid, and where the astronomical number 10^{30} ensures that at least two G^M projected sweeps are carried out),

$$\bar{F}^M(x) = F^M(x), \text{ for } x \in G^M,$$

and

(2.26)

$$\bar{U}^M(x) = U^M(x), \text{ for } x \in G^M.$$

We now make a number of G^M projected sweeps,

$$\bar{u}^M \leftarrow \text{Projected Gauss-Seidel } [\bar{u}^M; L^M, \bar{F}^M]. \quad (2.27)$$

After each sweep we test whether

$$\|\nabla \bar{u}^M\|_G \leq \epsilon^M. \quad (2.28)$$

If so, the accuracy criterion is satisfied, and we accept $\bar{u}^M$ as an accurate approximation to $U^M \equiv \bar{U}^M$ on G^M .

It is known that Gauss-Seidel iteration is a smoothing process: the error $\bar{U}^M(x) - \bar{u}^M(x)$ becomes smoother as the number of sweeps increases, while, at the same time, the rate of convergence slows down. We, therefore, carry out only a few G^M projected sweeps, stopping when either (2.28) is satisfied or

$$\|\nabla \bar{u}^M\|_G \geq \eta \|\nabla \bar{u}_{\text{old}}^M\|_G. \quad (2.29)$$

Here, η is a fixed parameter; in our work we have taken $\eta = .5$.

Suppose that (2.28) is not satisfied but that (2.29) is satisfied. This means on the one hand that the accuracy of $\bar{u}^M$ must be improved and on the other hand that it is inefficient to continue iterating on G^M . The slow rate of convergence on G^M indicates that the error is smooth, so that the error can be represented satisfactorily to the next coarsest grid, G^{M-1} . We therefore move to G^{M-1} .

Since $\bar{U}^M(x)$ satisfies (2.2), with $k = M$ and $F^M = \bar{F}^M$, the error

$$V^M(x) = \bar{U}^M(x) - \bar{u}^M(x), \quad (2.30)$$

satisfies the LCP

$$\begin{aligned} L^M V^M(x) &\leq \bar{F}^M(x), \quad \text{on } G^M, \\ V^M(x) + \bar{u}^M(x) &\geq 0, \quad \text{on } G^M, \\ \{V^M(x) + \bar{u}^M(x)\} [L^M V^M(x) - \bar{F}^M(x)] &= 0, \quad \text{on } G^M, \\ V^M(x) &= 0, \quad \text{on } \partial G^M, \end{aligned} \quad (2.31)$$

where the residual $\bar{r}^M$ is given by

$$\bar{r}^M(x) = \bar{F}^M(x) - L^M u^M(x), \quad x \in G^M. \quad (2.32)$$

As already observed, $V^M(x)$ is a smooth function and may, therefore, be accurately represented on G^{M-1} . Furthermore, comparing (2.31) and (1.1) we see that $V^M(x)$ is an approximation to the continuous solution $v(x)$ of the LCP

$$\begin{aligned} Lv(x) &\leq \bar{r}^M(x), \quad x \in \Omega, \\ v(x) + u^M(x) &\geq 0, \quad x \in \Omega, \\ [v(x) + u^M(x)][Lv(x) - \bar{r}^M(x)] &= 0 \quad x \in \Omega, \\ v(x) &= 0, \quad \text{on } \partial\Omega, \end{aligned} \quad (2.33)$$

(where, by abuse of notation, $\bar{r}^M(x)$ and $u^M(x)$ are defined on Ω by appropriate interpolation between the values of $\bar{r}^M$ and u^M on the gridpoints of G^M). Thus, a good approximation to $V^M(x)$ may be obtained by solving the finite difference approximation to (2.33) on G^{M-1} . That is, $V^M(x)$ is closely approximated on G^{M-1} by the solution $W^{M-1}(x)$ of the LCP,

$$\begin{aligned} (a) \quad L^{M-1} W^{M-1}(x) &\leq S_M^{M-1-M} \bar{r}^M(x), \quad \text{on } G^{M-1}, \\ (b) \quad W^{M-1}(x) + I_M^{M-1-M} u^M(x) &\geq 0, \quad \text{on } G^{M-1}, \\ (c) \quad [W^{M-1}(x) + I_M^{M-1-M} u^M(x)][L^{M-1} W^{M-1}(x) - S_M^{M-1-M} \bar{r}^M(x)] &= 0, \quad \text{on } G^{M-1}, \\ (d) \quad W^{M-1}(x) &= 0, \quad \text{on } \partial G^{M-1}. \end{aligned} \quad (2.34)$$

Here I_M^{M-1} and S_M^{M-1} are operators taking grid functions on G^M into grid functions on G^{M-1} . (As an aid in memorization, note that in $I_M^{M-1-M} u^M$ the subscript M and superscript M "cancel".)

The operators I_M^{M-1} and S_M^{M-1} can be defined in many ways. One choice is to choose both I_M^{M-1} and S_{M-1}^M to be the injection operator:

$$\text{Inj}_M^{M-1} w(x) = w(x), \quad x \in G^{M-1} \quad (2.35)$$

Other choices for I_M^{M-1} and S_M^{M-1} will be discussed later.

If we were solving a linear boundary value problem then condition (2.34b) would not apply and it would be most efficient to solve for the correction W^{M-1} on G^{M-1} . Since we are solving inequalities the problem is nonlinear and it is necessary to solve for a 'full approximation' $\bar{U}^{M-1}$ on G^{M-1} .

Setting

$$\bar{U}^{M-1}(x) = W^{M-1}(x) + I_M^{M-1} \bar{U}^M(x), \quad (2.36)$$

it follows that $\bar{U}^{M-1}(x)$ satisfies the LCP

$$\begin{aligned} (a) \quad & L^{M-1} \bar{U}^{M-1}(x) \leq \bar{F}^{M-1}(x), \quad \text{in } G^{M-1}, \\ (b) \quad & \bar{U}^{M-1}(x) \geq 0, \quad \text{in } G^{M-1}, \\ (c) \quad & \bar{U}^{M-1}(x) [L^{M-1} \bar{U}^{M-1}(x) - \bar{F}^{M-1}(x)] = 0, \quad \text{in } G^{M-1}, \\ (d) \quad & \bar{U}^{M-1}(x) = g(x), \quad \text{on } \partial G^{M-1}, \end{aligned} \quad (2.37)$$

where

$$\bar{F}^{M-1}(x) = S_M^{M-1} \bar{F}^M(x) + L^{M-1} I_M^{M-1} \bar{U}^M(x) = S_M^{M-1} [\bar{F}^M(x) - L^M \bar{U}^M(x)] + L^{M-1} I_M^{M-1} \bar{U}^M(x). \quad (2.38)$$

Finally, we set

$$\epsilon^{M-1} = \delta \|\nabla \bar{U}^M\|_G, \quad (2.39)$$

and

$$\bar{U}^{M-1} = I_M^{M-1} \bar{U}^M, \quad (2.40)$$

where δ is a constant; in our computations δ has been set equal to .15.

To recapitulate, starting with initial values of $\bar{U}^M$, ϵ^M , and $\bar{F}^M$, we first carry out G^M projected sweeps until convergence slows down. We then introduce a subsidiary problem on G^{M-1} with known $\bar{F}^{M-1}$ and ϵ^{M-1} and initial approximation $\bar{U}^{M-1}$. The process can be repeated, so that at any one stage of the computation we have a sequence of grid approximations $\bar{U}^M, \bar{U}^{M-1}, \dots, \bar{U}^{k-1}$, (approximating $\bar{U}^M, \bar{U}^{M-1}, \dots, \bar{U}^{k-1}$, respectively), tolerances $\epsilon^M, \epsilon^{M-1}, \dots, \epsilon^{k-1}$, and right hand sides $\bar{F}^M, \bar{F}^{M-1}, \dots, \bar{F}^{k-1}$.

In the general case, $\bar{u}^k$ is the solution of the LCP

$$\begin{aligned}
 (a) \quad & L \bar{u}^{k-k}(x) \leq \bar{F}^k(x), \quad \text{in } G^k, \\
 (b) \quad & \bar{u}^k(x) \geq 0, \quad \text{in } G^k, \\
 (c) \quad & \bar{u}^k(x) (L \bar{u}^{k-k}(x) - \bar{F}^k(x)) = 0, \quad \text{in } G^k, \\
 (d) \quad & \bar{u}^k(x) = g(x) \quad \text{on } \partial G^k;
 \end{aligned} \tag{2.41}$$

or equivalently,

$$\begin{aligned}
 (a) \quad & A \bar{u}^{k-k} \leq \bar{b}^k, \\
 (b) \quad & \bar{u}^k \geq 0, \\
 (c) \quad & (\bar{u}^k)^T (A \bar{u}^{k-k} - \bar{b}^k) = 0.
 \end{aligned} \tag{2.42}$$

This LCP is solved approximately using G^k projected sweeps until the latest approximation $\bar{u}^k$ satisfies either

$$\|\nabla \bar{u}^k\|_G \leq \epsilon^k, \tag{2.43}$$

or

$$\|\nabla \bar{u}^k\|_G \geq \eta \|\nabla \bar{u}_{old}^k\|_G. \tag{2.44}$$

If (2.44) holds but (2.43) does not then a new problem on G^{k-1} is defined by setting

$$\bar{F}^{k-1} = S_k^{k-1} [\bar{F}^k - L \bar{u}^{k-k}] + L^{k-1} I_k^{k-1-k} \bar{u}^k, \tag{2.45}$$

$$\epsilon^{k-1} = \delta \|\nabla \bar{u}^k\|_G, \tag{2.46}$$

$$\bar{u}^{k-1} = I_k^{k-1-k} \bar{u}^k, \tag{2.47}$$

$$\bar{u}^{k-1} = W^{k-1} + I_k^{k-1-k} \bar{u}^k, \tag{2.48}$$

$$v^k = \bar{u}^k - \bar{u}^k, \tag{2.49}$$

where W^{k-1} is an approximation to v^k on G^{k-1} . Unless otherwise indicated, I_k^{k-1} and S_k^{k-1} will be taken to be the injection operator Inj_k^{k-1} .

At some stage the latest approximation $\bar{u}^{k-1}$ must satisfy, (2.43)

$$\|\bar{u}^{k-1}\|_G \leq \epsilon^{k-1}, \tag{2.50}$$

if for no other reason than that when $k-1 = 1$ we cannot introduce any more subsidiary problems and must iterate until (2.50) is satisfied. Having found an approximation u^{k-1} of sufficient accuracy, we return to G^k . To do so, we first determine an approximation w^{k-1} to W^{k-1} from (2.48) namely

$$w^{k-1} = u^{k-1} - I_k^{k-1-k} u^k. \quad (2.51)$$

Next, let I_{k-1}^k be an interpolation operator taking grid functions on G^{k-1} into grid functions on G^k . A possible choice for I_{k-1}^k is the linear interpolation operator L_{k-1}^k defined as follows. If P_1, P_2, P_3 , and P_4 are the corners of a square in G^{k-1} (see Figure 2.1) then

$$L_{k-1}^k w^{k-1}(P_i) = \begin{cases} w^{k-1}(P_i), & 1 \leq i \leq 4, \\ (w^{k-1}(P_1) + w^{k-1}(P_2))/2, & i = 5, \\ (w^{k-1}(P_1) + w^{k-1}(P_4))/2, & i = 6, \\ (\sum_{i=1}^4 w^{k-1}(P_i))/4, & i = 7. \end{cases} \quad (2.52)$$

(Other choices for I_{k-1}^k will be discussed later.)

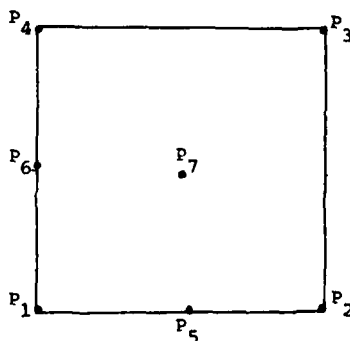


Figure 2.1: Linear interpolation from G^{k-1} to G^k .

Since w^{k-1} is an approximation to v^k on G^{k-1} ,

$$I_{k-1}^k w^{k-1} = I_{k-1}^k [u^{k-1} - I_k^{k-1-k} u^k], \quad (2.53)$$

is an approximation to v^k , and, noting (2.49),

$$\bar{u}^k = \bar{u}^k + I_{k-1}^k w^{k-1}, \quad (2.54)$$

is an improved approximation to $\bar{u}^k$. However, because of the nonnegativity constraint upon $\bar{u}^k$, we allow somewhat greater generality and replace $\bar{u}^k$ as follows:

$$\bar{u}^k + \varphi(\bar{u}^k; \bar{u}^k) = \varphi(\bar{u}^k + I_{k-1}^k w^{k-1}; \bar{u}^k). \quad (2.55)$$

Initially we set

$$\varphi(\bar{u}^k; \bar{u}^k) = \bar{u}^k, \quad (2.56)$$

but other choices will be considered later.

PFAS is described by (2.24) through (2.56). A flowchart is given in Figure 3.1, and the implementation is discussed in Section 3. If the algorithm converges, we will eventually obtain an approximation $\bar{u}^M$ satisfying the required accuracy condition (2.28) and the algorithm will terminate.

3. IMPLEMENTATION OF PFAS.

The flowchart for PFAS is given in Figure 3.1. PFAS has been implemented as a FORTRAN subroutine for the case when Ω is a rectangle in R^2 , $\mathcal{L}$ is the Laplacian operator, I_k^{k-1} and S_k^{k-1} are injections (equation (2.35)), and I_{k-1}^k is linear interpolation (equation (2.52)). The subroutine PFAS, which is listed in Appendix A as part of the program for solving the porous flow free boundary problem described in Section 4, is a straightforward modification of an earlier program, FAS Cycle C, of Brandt. In the subroutine PFAS most of the computations are performed by auxiliary subroutines, and the flowchart shows the role played by these auxiliary subroutines.

One reason for giving a listing of PFAS is so that the reader can appreciate how easy it is to implement PFAS. It may also be remarked that many other interesting free boundary problems (for example, elastic-plastic torsion problems and cavitating journal bearing problems) are formulated on simple polygonal regions, and the program given here could easily be modified to handle these problems.

The following comments arise:

1. In PFAS, the LCP for $\bar{u}^k$ is solved in the form (2.42) rather than (2.41), but the values of $\bar{u}^k$ on ∂G^k are also stored. Thus, $\bar{b}^k = h_k^{2-k} \bar{F}^k$ is stored instead of $\bar{F}^k$. In going from G^k to G^{k-1} we have, from (2.45), since $h_{k-1} = 2h_k$,

$$\begin{aligned} \bar{b}^{k-1} &= h_{k-1}^2 \bar{F}^{k-1}, \\ &= h_{k-1}^2 (S_k^{k-1} [\bar{F}^k - L_k^{k-k} \bar{u}^k] + L^{k-1} I_k^{k-1} \bar{u}^k), \\ &= h_{k-1}^2 (S_k^{k-1} h_k^{-2} [\bar{b}^k - A_k^{k-k} \bar{u}^k] + L^{k-1} I_k^{k-1} \bar{u}^k), \\ &= 4 S_k^{k-1} [\bar{b}^k - A_k^{k-k} \bar{u}^k] + A^{k-1} I_k^{k-1} \bar{u}^k. \end{aligned} \quad (3.1)$$

2. A G^k -work-unit is the work required for one G^k projected sweep. The work for one G^k projected sweep is approximately $2^{-n(M-k)} G^M$ -work-units, and PFAS keeps track of the total number of G^M -work-units, WU. When no confusion is possible we write "work unit" instead of " G^M -work-unit".

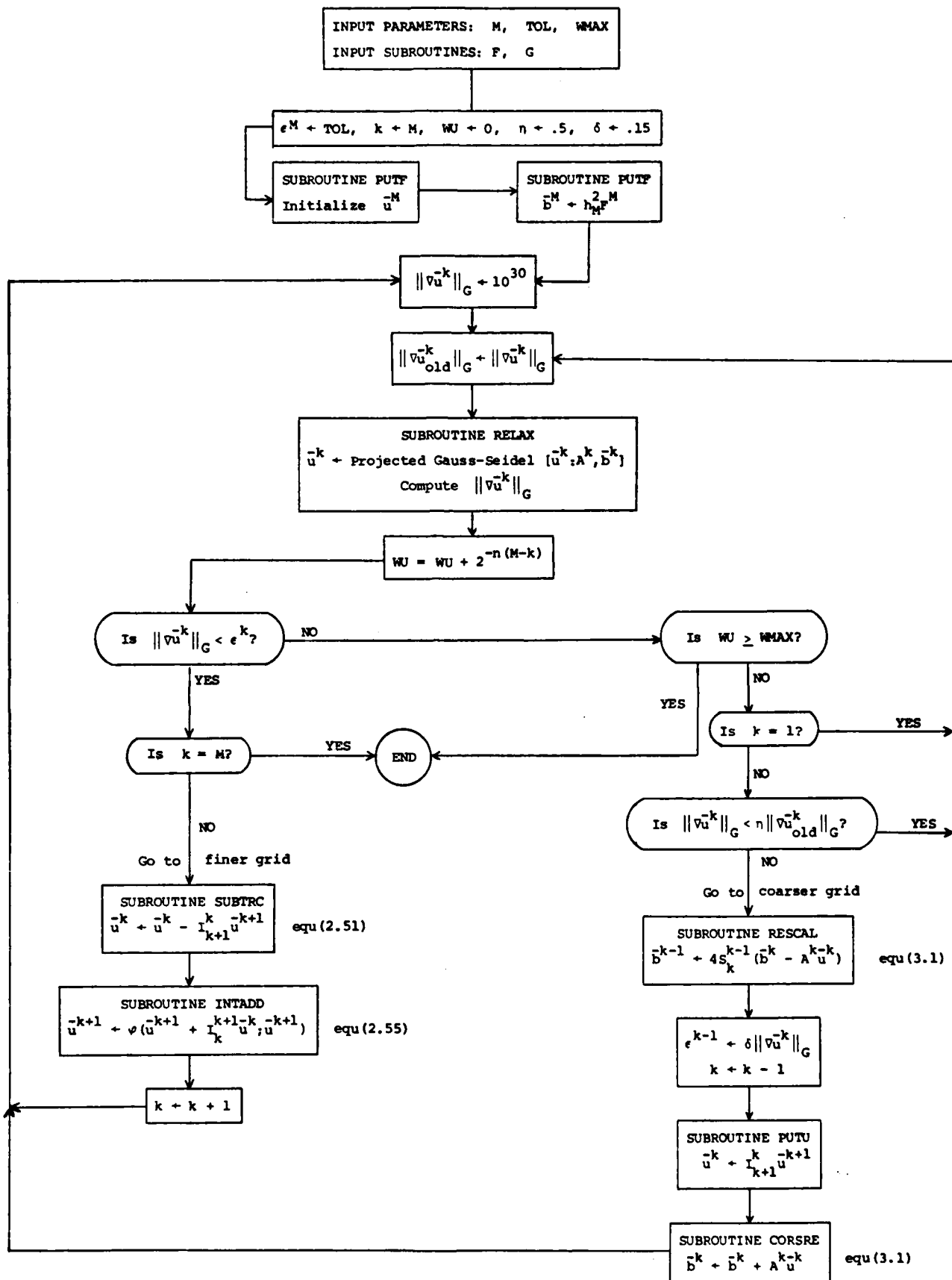


Figure 3.1: Flow Chart for PFAS.

3. The asymptotic speed of convergence is measured by the asymptotic convergence factor $\overset{\circ}{\mu}$, which is defined by

$$\overset{\circ}{\mu} = \lim_{WU \rightarrow \infty} [\|\nabla u^{-M}\|_G]^{1/WU} \quad (3.2)$$

4. All the numerical computations were performed on the Univac 1180 at the University of Wisconsin-Madison. The programs were written in ASCII FORTRAN and compiled and executed using full optimization.

The Univac 1180 single-precision arithmetic has approximately eight decimals. The residuals usually decrease quite rapidly at the beginning of a computation so the round-off threshold is quickly reached. For example, for the problem considered in Section 4 with $M = 5$, $\|u^M\|_G$ is about 2×10^3 and the single precision algorithm went into a loop when $\|\nabla u^{-M}\|_G$ reached 5×10^{-6} after a mere 50 work units.

In the numerical experiments we were particularly interested in measuring the asymptotic convergence factor $\overset{\circ}{\mu}$. To eliminate round-off effects, all the computations reported on here used double precision arithmetic. Of course, this is not normally necessary. Furthermore, even if very accurate solutions of the discrete problem (2.2) were required, it would suffice to store u^{-M} in double precision and all other quantities in single precision.

The execution times quoted are those provided by the Univac 1180 Exec. System. As is often the case on timesharing systems, the times are only reproducible to within about 10%.

Because of its word length, the UNIVAC 1180 can only directly access 64K words of storage. When $M \geq 7$ more than 64K words of storage are needed by PFAS and there is a significant degradation in performance.

5. To measure $\overset{\circ}{\mu}$ the iterations were continued for the first 100 work units, unless the residuals vanished before. In practice one usually iterates only for about 30 work units.

We also used several values of M in order to measure the dependence of $\overset{\circ}{\mu}$ upon M .

Part of the output of a typical computation using PFAS is shown in Figure 3.2. After each G^k projected sweep, the values of the level k , the residual norm $\|\bar{v}_u^k\|_G$, and the number of work units WU are printed out.

The computations starting at a level M /level $(M-1)$ junction and continuing until the next level M /level $(M-1)$ junction are called a cycle (see Figure 3.2). For the cycle shown in Figure 3.2, $\|\bar{v}_u^M\|_G$ decreased from $.293 \cdot 10^{-9}$ to $.110 \cdot 10^{-9}$ with the expenditure of $(99.039-94.400) = 4.639$ work units.

While minor variations do arise, a cycle often consists of a sequence of 2 sweeps at each of levels $M-1, M-2, \dots, 1$, followed by 2 sweeps at each of levels $2, \dots, M-1$, terminating with 2 or 3 sweeps at level M . If this pattern is followed with 3 sweeps at level M then the average number of work units per cycle is

$$3 + 4[2^{-n} + 2^{-2n} + \dots] = 3 + 4/(2^n - 1), \quad (3.3)$$

and the average number of work units per G^M projected sweep is $1 + 4/(3(2^n - 1))$.

Of course, very irregular patterns are observed when the round-off threshold is reached.

6. As can be seen from Figure 3.2, $\|\bar{v}_u^M\|_G$ decreases steadily but not very regularly, in part because of slight variations in the number of sweeps at each level. To evaluate the algorithm, we have used two quantities:

$$r_f = \|\bar{v}_{u_{\text{final}}}^M\|_G = \text{the value of } \|\bar{v}_u^M\|_G \text{ at the end of the last complete cycle before 100 work units,} \quad (3.4)$$

$$\mu_f = [\|\bar{v}_{u_{\text{final}}}^M\|_G / \|\bar{v}_{u_{\text{initial}}}^M\|_G]^{1/[WU_{\text{final}} - WU_{\text{initial}}]}, \quad (3.5)$$

where $\|\bar{v}_{u_{\text{initial}}}^M\|_G$ is the value of $\|\bar{v}_u^M\|_G$ after the first G^M sweep. μ_f is an estimate for the asymptotic convergence factor μ .

For example, for the data in Figure 3.2, the value of $\|\bar{v}_{u_{\text{initial}}}^M\|_G$ (which is not shown in Figure 3.2) was 4.95 and, of course, $WU_{\text{initial}} = 1$. Thus,

```

LEVEL 5   RESIDUAL NORM= .755-010   WORK= 91.400
LEVEL 6   RESIDUAL NORM= .126-008   WORK= 92.400
LEVEL 6   RESIDUAL NORM= .515-009   WORK= 93.400
LEVEL 6   RESIDUAL NORM= .293-009   WORK= 94.400
*****END OF CYCLE*****MU = .7771
LEVEL 5   RESIDUAL NORM= .196-009   WORK= 94.650
LEVEL 5   RESIDUAL NORM= .133-009   WORK= 94.900
LEVEL 4   RESIDUAL NORM= .879-010   WORK= 94.963
LEVEL 4   RESIDUAL NORM= .613-010   WORK= 95.025
LEVEL 3   RESIDUAL NORM= .385-010   WORK= 95.041
LEVEL 3   RESIDUAL NORM= .257-010   WORK= 95.057
LEVEL 2   RESIDUAL NORM= .133-010   WORK= 95.061
LEVEL 2   RESIDUAL NORM= .717-011   WORK= 95.064
LEVEL 1   RESIDUAL NORM= .243-011   WORK= 95.065
LEVEL 1   RESIDUAL NORM= .447-012   WORK= 95.066
LEVEL 2   RESIDUAL NORM= .303-011   WORK= 95.070
LEVEL 3   RESIDUAL NORM= .189-010   WORK= 95.086
LEVEL 3   RESIDUAL NORM= .714-011   WORK= 95.102
LEVEL 4   RESIDUAL NORM= .686-010   WORK= 95.164
LEVEL 4   RESIDUAL NORM= .255-010   WORK= 95.227
LEVEL 4   RESIDUAL NORM= .138-010   WORK= 95.289
LEVEL 5   RESIDUAL NORM= .151-009   WORK= 95.539
LEVEL 5   RESIDUAL NORM= .534-010   WORK= 95.789
LEVEL 5   RESIDUAL NORM= .284-010   WORK= 96.039
LEVEL 6   RESIDUAL NORM= .473-009   WORK= 97.039
LEVEL 6   RESIDUAL NORM= .194-009   WORK= 98.039
LEVEL 6   RESIDUAL NORM= .110-009   WORK= 99.039
*****END OF CYCLE*****MU = .7787
LEVEL 5   RESIDUAL NORM= .737-010   WORK= 99.289
LEVEL 5   RESIDUAL NORM= .499-010   WORK= 99.539
LEVEL 4   RESIDUAL NORM= .331-010   WORK= 99.602
LEVEL 4   RESIDUAL NORM= .231-010   WORK= 99.664
LEVEL 3   RESIDUAL NORM= .145-010   WORK= 99.680

```

Figure 3.2: Typical Output for the PFAS Algorithm.
(M = 6, Problem (4.1)-(4.2), Run #X67368)

$$r_f \doteq .110 \cdot 10^{-9} ,$$

and

$$\dot{\mu}_f = \left[\frac{.110 \cdot 10^{-9}}{4.95} \right]^{1/(99.039-1)} \doteq .7787 .$$

We usually only quote r_f to one decimal place and $\dot{\mu}_f$ to two decimal places, since this is quite adequate for our purposes.

PFAS computes and prints

$$\dot{\mu} = \left[\frac{\|\nabla u^M\|_G}{\|\nabla u^M_{\text{initial}}\|_G} \right]^{1/[WU-WU_{\text{initial}}]} \quad (3.6)$$

at the end of each cycle.

7. In all the experiments reported here the parameters δ and η (see (2.29) and (2.39)) were given by $\delta = .5$ and $\eta = .15$. According to Brandt [77] the rate of convergence is not very sensitive to changes in these parameters, and this was confirmed in a few experiments.

In a few cases, but never for $\delta = .5$ and $\eta = .15$, the program "hunted": that is, the program went down from G^M to G^1 , up to G^k for $k < M$, and then down again to G^1 instead of continuing up to G^M . This might happen several times before G^M was reached again.

4. NUMERICAL RESULTS FOR POROUS FLOW THROUGH A DAM.

Calculations were performed on the well-known free boundary problem describing the flow of water through a porous dam. The geometry is shown in Figure 4.1. Water seeps from a reservoir of height y_1 through a rectangular dam of width a to a reservoir of height y_2 . Part of the dam is saturated and the remainder of the dam is dry. The wet and dry regions are separated by an unknown free boundary which must be found as part of the solution. For an introduction to the problem see Bear [1972], or Cryer [1976].

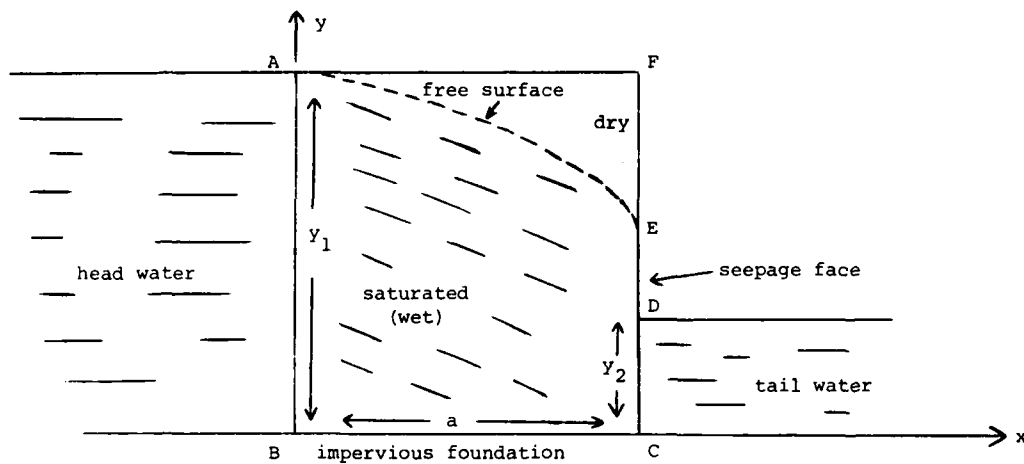


Figure 4.1 Seepage Through a Simple Rectangular Dam

As shown by Baiocchi [1971] the problem can be formulated as follows: Find u on the rectangle $\Omega = ABCF$ such that

$$\begin{aligned} \nabla^2 u &\leq 1, & \text{on } \Omega, \\ u &\geq 0, & \text{on } \Omega, \\ u(\nabla^2 u - 1) &= 0, & \text{on } \Omega, \end{aligned} \quad (4.1)$$

$$u = g = \begin{cases} (y_1 - y)^2/2 & \text{on AB,} \\ (y_2 - y)^2/2 & \text{on CD,} \\ [y_1^2(a - x) + y_2^2(x)]/2a, & \text{on BC,} \\ 0, & \text{on DFA,} \end{cases} \quad (4.2)$$

which is in the form (1.1).

This problem was solved using PFAS, with I_k^{k-1} and S_k^{k-1} being injections (equation (2.35)) and with I_{k-1}^k defined by linear interpolation (equation (2.52)). The initial values of $\bar{u}^M$ were obtained by interpolating the boundary values of u linearly in the x direction. A listing of the program is given in Appendix A.

We considered the well-known case, $y_1 = 24$, $y_2 = 4$, and $a = 16$. In all computations G^1 was a $(2 + 1) \times (3 + 1)$ grid with $h_1 = 8$. The finest grid used was G^7 with $(128 + 1) \times (192 + 1) = 24897$ grid points.

To give the reader an idea of the solution, the solution U^2 of (2.2) is given to four decimal places in Table 4.1.

$\begin{smallmatrix} x \\ y \end{smallmatrix}$	0	4	8	12	16
24	0	0	0	0	0
20	8	2.5371	0	0	0
16	32	18.1486	6.7841	0	0
12	72	47.2732	24.9879	7.9120	0
8	128	89.9564	53.9823	22.6601	0
4	200	146.5702	94.3247	44.7462	0
0	288	218.0000	148.0000	78.0000	8

Table 4.1. U^2 for the Dam Problem
(Run #X34654)

The numerical results, for different values of M , and $\epsilon^M = \text{TOL} = 0$, are given in Table 4.2. The most important conclusions are that convergence always occurred and that the convergence factor $\bar{\nu}_f$ is always less than .81.

Run #	X34654	X34654	X34654	X34654	X34654	PC 3567
M	2	3	4	5	6	7
G^M	5×7	9×13	17×25	33×49	65×97	129×193
r_f	0*	4(-17)*	1(-13)	1(-8)	1(-10)	1(-7)
ρ_f	.404	.607	.726	.813	.778	.81
Execution Time for 100 Work Units (Seconds)	.114	.428	1.04	3.55	13.39	**
ρ_{SORopt}	.18	.49	.71	.84	.92	.96

Table 4.2. Solution of the dam problem using PFAS

*Reached round-off level before 100 work units.

**Required 70K workspace so extended storage facility invoked, and timing not compatible.

We now compare the convergence factors ρ_f in Table 4.2 with those for other methods of solving the LCP (2.2).

A popular method of solving the LCP (1.3) is G^M projected SOR (point SOR with projection) which has also been called "modified SOR" by Cottle.

When using G^M projected SOR it is observed experimentally that the values of u^M settle down quite quickly into positive values and zero values. Thereafter G^M projected SOR is equivalent to using point SOR on the subset $G_+^M = \{x \in G^M : u^M(x) > 0\}$. Thus the asymptotic convergence factor for G^M projected SOR is in general equal to the asymptotic convergence factor for point SOR on G_+^M . It is known (Varga [1962, p. 294]) that for a region of area A and for the finite difference equations corresponding to the five-point difference approximation to Laplace's equation with stepsize h , the convergence factor for the optimum choice of overrelaxation parameter ω is approximated quite well by

$$\rho_1(h) = \frac{2}{1 + 3.015[h^2/A]^{1/2}} - 1. \quad (4.3)$$

In the present case we do not know the area of G_+^M , but, as a rough guide, the area of G_+^M is approximately equal to the area of Ω , which is about 80% of the area of the rectangle ABCF. Therefore, for our present purposes the asymptotic convergence factor for G^M projected SOR with optimum choice of ω may be taken to be

$$\rho_{\text{SORopt}} \doteq \frac{2}{1 + 3.015[h^2/(.8 \times 16 \times 24)]^{1/2}} - 1 \doteq \frac{2}{1 + .172 h} - 1, \quad (4.4)$$

and these values are given at the bottom of Table 4.2.

As Table 4.2 shows, for large problems, PFAS is faster than G^M projected SOR. On G^7 , for example, the increase in speed (measured in work units) is $\ln.96/\ln.81 \doteq 5.2$. Against this, two factors must be borne in mind: (1) PFAS is more complicated and requires more overhead per work unit; (2) PFAS requires somewhat more storage. We discuss these two factors below, but before doing so we wish to emphasize that although these factors reduce the advantage in speed of PFAS, the measured execution times for PFAS are much smaller than those for G^M projected SOR (see Tables 5.3 and 6.3).

1. Overhead.

To obtain an indication of the additional overhead required by PFAS, we compared execution times for $M = 5$. We first used PFAS with $\epsilon^M = 2.10^{-8}$. This required 96.156 work units and took 3.40 seconds. We then modified PFAS so that only the grid $k = M$ was used and so that over-relaxation was used with the over-relaxation parameter ω given by equation (4.4). We were thus using G^M projected SOR with a nearly optimum ω . To reduce $\|\nabla u^M\|_G$ to $\epsilon^M = 2.10^{-8}$ required 146 work units and took 4.82 seconds. Since

$$(3.40/96.156)/(4.82/146) \doteq 1.07$$

we conclude that, in this application, the additional overhead required by PFAS only increases the computation time per G^M work unit by about 10%.

2. Storage.

As implemented here, PFAS keeps the solutions and residuals on all the grids, and therefore requires storage for $2[1 + 4^{-1} + 4^{-2} + \dots] = 8/3 G^M$ grids. In contrast, G^M projected SOR requires storage for only one G^M grid.

If storage is at a premium, the residuals on G^M need not be stored and PFAS requires only 5/3 times as much storage as G^M projected SOR. If $\bar{u}^M$ is stored to double precision, but $\bar{u}^k$ and $\bar{b}^k$ are stored to single precision for $k < M$, only 4/3 times as much storage is needed. If $F(x)$ were not the constant 1, but a complicated function, then either the function values or the residuals would have to be stored for G^M projected SOR, and PFAS would require at most 33% more storage. Finally, the PFMG algorithm described in Section 6 often need not store any data on the G^M grid (see Section 6).

Another possible algorithm for solving the LCP (1.3) is the MBSOR (modified block SOR) algorithm of Cottle and Sacher [1978]. This algorithm is based upon the solution of a sequence of "one-dimensional" LCP's in much the same way that line SOR is based upon solving a sequence of "one-dimensional" equations. We used MBSOR to solve the dam problem (4.1), (4.2), for the case $M = 5$. The program was kindly provided by Professor Sacher. We tried a few values of the over-relaxation parameter ω , and found that 1.8 gave the best results. With $\omega = 1.8$ MBSOR required 114 iterations to reduce $\| \nabla u^M \|_G$ to below 2.10^{-8} and took 13.13 seconds. The following comments arise:

1. In numerical experiments on the dam problem, Cottle [1974] found that MBSOR was about 20% faster than "modified point SOR", that is, G^M projected SOR. This is consistent with the fact that, for equations, the convergence ratio for line SOR is only faster by a factor of $\sqrt{2}$ than point SOR while there is more computation per iteration. This is also consistent with the present results, since G^M projected SOR required 146 iterations to reduce the residual to 2.10^{-8} while MBSOR required only 114.

2. The poor execution time of MBSOR (13.13 seconds) compared to PFAS (3.40 seconds) can be explained in part by two factors: (a) MBSOR requires more computation per iteration than is needed by PFAS for a single work unit; (b) the MBSOR program was

written for the case of general coefficients, while the PFAS program takes advantage of the properties of the five-point difference operator.

3. It must also be borne in mind that Cottle and Sacher [1978] found that MBSOR was three times as fast as G^M projected SOR for the journal bearing problem where the solution is zero at a high percentage of the gridpoints.

We conclude from Table 4.2 and from the above discussion, that for the dam problem (4.1), (4.2) PFAS is faster than G^M projected SOR and modified block SOR for $M \geq 5$, that is, for grids of dimension 33×49 or greater. Furthermore, we also conclude that the values of $\bar{\mu}_f$ and ρ_{SORopt} in Table 4.2 provide a reasonably accurate guide to the relative performance of PFAS and G^M projected SOR. We believe that PFAS will be faster than both G^M projected SOR and MBSOR for a wide range of problems.

4. For a grid G^M with N gridpoints, both G^M -projected SOR and modified block SOR have computation times which are $O(N^{3/2})$. As Table 4.2 shows, the computation time for PFAS is $O(N)$. Therefore, the performance of PFAS vis-a-vis the other methods improves as the grids become finer.

5. ALTERNATIVE IMPLEMENTATIONS OF PFAS.

In this section we discuss alternative implementations of PFAS, the best of which achieves substantially improved performance.

The improvement in PFAS which might be possible is suggested by considering the asymptotic convergence ratio, $\bar{\mu}_{\text{FAS}}$ say, for FAS for Poisson's equation. For FAS, the error reduction per G^M -sweep is .5. If each G^M -sweep is accompanied by, on average, one G^k sweep for $1 \leq k \leq M-1$, then the number of work units per G^M -sweep is

$$1 + 2^{-2} + 2^{-4} + \dots = 4/3$$

and the convergence ratio is $(.5)^{3/4} = .595$, as stated by Brandt [1977, p. 351]. In the present case, as observed in Section 3, the average number of work units per G^M sweep is

$$1 + 4/[3(2^n - 1)] = 13/9 ,$$

so that

$$\bar{\mu}_{\text{FAS}} = (.5)^{9/13} = .6188 . \quad (5.1)$$

This value of $\bar{\mu}_{\text{FAS}}$ is observed experimentally. The worst observed value of $\bar{\mu}_f$ for the PFAS results quoted in Section 3 was $\bar{\mu}_f = .81$. Thus, FAS (for equations) is faster than PFAS (for LCP's) by a factor of $\ln .81 / \ln .6188 = 2.28$.

Plausible reasons why PFAS is slower than FAS include the following:

D1: Negative components of $\bar{u}^k$.

The inequality (2.41b) requires that $\bar{u}^k$ be non-negative. In each G^k projected sweep the step (2.7) ensures that $\bar{u}^k$ is non-negative. Furthermore, if I_k^{k-1} is the injection operator the initial approximation $\bar{u}^{k-1}$ defined by (2.47) is also non-negative. However, (2.54) does not preserve non-negativity: in returning to G^k from G^{k-1} the initial approximation $\bar{u}^k$ may have negative components, and this is often observed. Of course, any negative components are removed in the first subsequent G^k projected sweep, but nevertheless the introduction of negative components must retard convergence. □

D2: Large residuals near the free boundary.

At a point $x \in G^k$ where $\bar{U}^k(x) = 0$ the corresponding residual

$$\bar{R}^k(x) = \bar{F}^k(x) - L^k \bar{U}^k(x) \quad (5.2)$$

must be non-negative because of the inequality (2.41a) but need not be small. $\square$

D3: Influence of the discrete interface.

The discrete interface $\Gamma^k \subset R^2$ is the interface between the set of points where $\bar{U}^k > 0$ and the set of points where $\bar{U}^k = 0$. Γ^k approximates the continuous interface, or free boundary, Γ separating the points where the solution $u(x)$ is positive from the points where $u(x)$ is zero.

In special cases it may happen that $\Gamma^k = \Gamma$ for all k , in which case PFAS converges as fast as FAS. An example is given by problem (5.3), (5.4) below with $R = 2$, for which Γ is the line $y = 5 - 2x$; it is found experimentally that $\Gamma^k = \Gamma$ for $k \leq 6$.

In general, Γ^k and Γ differ by $O(h_k)$, and Γ^k and Γ^{k-1} differ by $O(h_k)$. In particular, it may happen that $\bar{U}^k(x) > 0$ while $\bar{U}^{k-1}(x) = 0$. Furthermore, near Γ^{k-1} the residuals may be less smooth because of the projection (2.7) and because of the irregular shape of Γ^k and Γ^{k-1} . This introduces errors in the coarse grid corrections (2.55) thereby slowing the rate of convergence. Finally, the injection operator (2.35) is not adequate if the data to which it is applied is not smooth. $\square$

Multigrid algorithms can often be speeded up by modifying the operators I_k^{k-1} , S_k^{k-1} , and I_{k-1}^k . We have tried a number of modifications of the corresponding PFAS subroutines which were intended to address the difficulties D1 to D3 mentioned above.

The subroutine PFAS in Appendix A was modified so as to facilitate experimentation. This was done by changing the calls to the auxiliary subroutines so that input "switch" parameters determined which version of each subroutine was used.

In addition, computations were also made for the following problem:

$$\begin{aligned} (a) \quad & \nabla^2 u \leq f(x,y), \quad \text{in } \Omega, \\ (b) \quad & u \geq 0, \quad \text{in } \Omega, \\ (c) \quad & u = g, \quad \text{on } \partial\Omega, \end{aligned} \quad (5.3)$$

where $\Omega = [0,3] \times [0,2]$, and where f and g are chosen so that the exact solution is

$$u = [\cos(x+y) + 2][\max\{0, 2.5R - Rx - y\}]^2. \quad (5.4)$$

Here, R is a parameter which is chosen close to the value 2. Note that $u \in C^2(\Omega)$ and $u = 0$ above the line $y = R(2.5 - x)$. By changing the value of R we can force gridpoints to lie very close to the exact free boundary; this may be expected to cause PFAS difficulty because if $\bar{u}^k(x)$ is positive but very small for some $x \in G^k$ then it will take PFAS a large number of iterations to determine whether $\bar{u}^k(x)$ is zero or positive.

The modified version of PFAS is called PFASMD and is listed in Appendix B as part of a program for solving the porous flow problem of Section 4 and problems (5.3), (5.4). PFASMD was used to compute all the results in this section.

Our first modifications to the auxiliary subroutines of PFAS were not very successful, but they were very instructive and we briefly summarize them. In all cases, the results are for the dam problem with $M = 5$. (All were with run #X35519).

M1. PFAS was modified so as to enforce nonnegativity of $\bar{u}^k$ immediately after returning from G^{k-1} . This was done by defining ϕ in (2.55) by

$$\phi(\bar{u}^k; \bar{u}^k) = \max\{0, \bar{u}^k\}. \quad (5.5)$$

The new subroutine was called INTAPR.

This modification converged slightly faster than PFAS with $\bar{u}_f = .803$. $\square$

M2. The usual situation in which the nonnegativity of $\bar{u}^k$ is violated is as follows.

Let $\bar{u}^k(x) = 0$, where $x \in G^k$ but $x \notin G^{k-1}$. Let $y \in G^{k-1}$ be a neighbor of x , such that $\bar{u}^k(y) > 0$. It may then happen that $w^{k-1}(y) < 0$. As a result, $(I_{k-1}^k w^{k-1})(x)$ may be negative, and if so the updated value of $\bar{u}^k(x)$ will be negative.

To avoid this, PFAS was modified by changing the subroutines SUBTRC and PUTU so that the operator I_k^{k-1} became

$$I_k^{k-1} u(y) = \begin{cases} \bar{u}^k(y) & \text{if } \bar{u}^k(x) > 0 \text{ for all eight} \\ & \text{neighbors } x \text{ of } y \text{ in } G^k, \\ 0, & \text{otherwise.} \end{cases} \quad (5.6)$$

The new subroutines were called PUTUNN and SUBTNN, respectively.

Remembering from (2.48) that

$$\bar{u}^{k-1} = w^{k-1} + I_k^{k-1} u^k,$$

we see from (5.6) and (2.41b) that the restraint $w^{k-1}(y) \geq 0$ is enforced for every point $y \in G^{k-1}$ with a neighbor $x \in G^k$ such that $\bar{u}^k(x) = 0$.

This modification converged slightly more slowly than PFAS, with $\bar{u}_f^0 = .817$. $\square$

M3. PFAS was modified so that if the current value of $\bar{u}^M(x)$ was zero, then $\bar{u}^k(x)$ was forced to be zero for $k < M$. This was done by changing the subroutine RELAX. In effect, (2.7) was followed by a further operation:

$$\text{If } k < M \text{ and } \bar{u}^M(x_j^k) = 0 \text{ then } \bar{u}_j^{k,s} = 0. \quad (5.7)$$

The new subroutine was called RELXFR.

This modification converged but much more slowly than PFAS with $\bar{u}_f^0 = .887$. $\square$

M4. Brandt [1977, p. 378] has found residual weighting useful when the coefficients of the differential equation are changing rapidly. We, therefore, changed the subroutine RESCAL so that S_k^{k-1} became:

$$4 S_k^{k-1} r^k(x) = \sum_{\Delta} \rho(\Delta) r^k(x + \Delta h_k), \quad (5.8)$$

where $\Delta = (\Delta_1, \Delta_2)$ for integers Δ_1, Δ_2 and the only nonzero $\rho(\Delta)$ are

$$\begin{aligned} \rho(0,0) &= 1, \\ \rho(0,1) &= \rho(1,0) = \rho(0,-1) = \rho(-1,0) = \frac{1}{2}, \\ \rho(1,1) &= \rho(1,-1) = \rho(-1,1) = \rho(-1,-1) = \frac{1}{4}. \end{aligned} \quad (5.9)$$

The new subroutine was called RESCAV.

This modification cycled between G^1 and G^2 , as did also the further modification for which I_k^{k-1} was also defined by (5.8), (5.9).

The nonconvergence of the modification M4 requires explanation, and this is provided by

Lemma 5.1.

Let φ be defined by (2.56). For $1 \leq k \leq M$ let $\bar{U}^k$ be the solution of the LCP (2.41), where $\bar{F}^k$ satisfies (2.45). Finally, let I_{k-1}^k satisfy

$$(I_{k-1}^k(z^{k-1}) = 0) \Rightarrow (z^{k-1} = 0), \text{ for all } z^{k-1} \in R^{N_{k-1}}. \quad (5.10)$$

Then for PFAS to converge it is necessary that

$$S_k^{k-1}[\bar{F}^k - L^k \bar{U}^k] \geq 0, \quad (5.11)$$

$$I_k^{k-1} \bar{U}^k \geq 0, \quad (5.12)$$

$$[I_k^{k-1} \bar{U}^k]^T S_k^{k-1} [\bar{F}^k - L^k \bar{U}^k] = 0. \quad (5.13)$$

Proof: We apply PFAS by setting $\bar{u}^k = \bar{U}^k$, and forming the LCP (2.41) on G^{k-1} :

$$L^{k-1} \bar{U}^{k-1} \leq \bar{F}^{k-1},$$

$$\bar{U}^{k-1} \geq 0, \quad (*)$$

$$(\bar{U}^{k-1})^T (L^{k-1} \bar{U}^{k-1} - \bar{F}^{k-1}) = 0.$$

Solving this exactly so that $\bar{u}^{k-1} = \bar{U}^{k-1}$, we then return to G^k . Since PFAS converges, the new value of $\bar{u}^k$ given by (2.55) must be equal to $\bar{U}^k$. That is,

$$I_{k-1}^k \bar{u}^{k-1} = I_{k-1}^k [\bar{U}^{k-1} - I_k^{k-1} \bar{U}^k] = 0,$$

which, from (5.10), implies that

$$\bar{U}^{k-1} = I_k^{k-1} \bar{U}^k.$$

Substituting into (*) and noting (2.45) we obtain (5.11) through (5.13). $\square$

The following remarks follow from Lemma 5.1.

1. Lemma 5.1 brings out an interesting difference between multigrid methods for equations and for inequalities. For equations, $\bar{F}^k - L^k \bar{U}^k = 0$ and conditions (5.11)-(5.13) are satisfied for any reasonable choice of S_k^{k-1} and I_k^{k-1} , but this is not true for inequalities. $\square$

2. Since $\bar{U}^k$ solves (2.41), inequalities (5.11) and (5.12) will certainly hold if S_k^{k-1} and I_k^{k-1} map nonnegative vectors into nonnegative vectors. In particular, this will be the case if S_k^{k-1} and I_k^{k-1} take linear combinations of values with nonnegative weights. $\square$

3. If S_k^{k-1} and I_k^{k-1} are injections, then (5.13) is implied by (2.41c). $\square$

4. If S_k^{k-1} is defined by (5.8) and (5.9) while I_k^{k-1} is injection then (5.13) does not hold in general. This is because in general there will be points $x, y \in G^k$ such that $x \in G^{k-1}$, $\bar{U}^k(x) > 0$, $\bar{U}^k(y) = 0$, y is a neighbor of x in G^k and $(\bar{F}^k - L^k \bar{U}^k)(y) > 0$. Then

$$I_k^{k-1} \bar{U}^k(x) = \bar{U}^k(x) > 0,$$

and

$$(S_k^{k-1}(\bar{F}^k - L^k \bar{U}^k))(x) \geq \frac{1}{4} (\bar{F}^k - L^k \bar{U}^k)(y) > 0,$$

so that (5.13) does not hold. This explains why the modification M4 of PFAS did not converge. $\square$

We now describe two further modifications of PFAS which were tried:

M5. Bearing Lemma 5.1 in mind it is possible to introduce weighted sums for which (5.13) does hold. One choice uses weighted residuals only near the boundary:

$$4S_k^{k-1} r^k(x) = \begin{cases} 4r^k(x), & \text{if } \bar{u}^k(x) = 0 \text{ or if } \bar{u}^k(y) > 0 \\ & \text{for all eight neighbors } y \in G^k \text{ of } x, \\ \sum_{\Delta} \rho(\Delta) r^k(x + \Delta h_k) \text{ signum } \{\bar{u}^k(x + \Delta h_k)\}, & \\ \text{otherwise} \end{cases} \quad (5.14)$$

where

$$\text{signum } \alpha = \begin{cases} 1, & \text{if } \alpha > 0, \\ 0, & \text{if } \alpha = 0, \end{cases}$$

and where the weights $\rho(\Delta)$ are as in (5.9). This was done by an appropriate change in the subroutine RESCAL; the new subroutine was called RESCL1.

M6. As mentioned in D1 and D3 above, if $\bar{u}^k(x) = 0$ then it may happen that $\bar{u}^k(x) = \bar{u}^k(x) + I_{k-1}^k w^{k-1}(x)$ is not zero. It can be argued that changes of $\bar{u}^k(x)$ from or to zero should only be done on G^k . We, therefore, modified the subroutine INTADD so that in (2.55) φ was defined by

$$\varphi(\bar{u}^k(x); \bar{u}^k(x)) = \begin{cases} \bar{u}^k(x), & \text{if } \bar{u}^k(x) > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (5.15)$$

The new subroutine was called INTADM. □

The modifications M5 and M6 are independent, and we solved (4.1), (4.2) with $M = 5$ and different combinations of M5 and M6. In each case, the computations were terminated when $\|\nabla \bar{u}^M\|_G \leq 2 \cdot 10^{-8}$. The results are summarized in Table 5.1.

Modifications	-	M5	M6	M5 and M6
Work Units	96.15	126.12	42.81	43.76
Execution Time (Seconds)	3.40	4.67	1.63	1.76
$\bar{u}_f$	.815	.854	.623	.623

Table 5.1: Solution of (4.1), (4.2) with $M = 5$ and $\epsilon^M = 2 \cdot 10^{-8}$ for modifications 5 and 6.
(Run #X35026)

The performance of PFAS is of course problem dependent. In Table 5.2 we compare modifications 5 and 6 for the problem (5.3), (5.4). As in Table 5.1 we iterated until $\|\nabla u^{(k)}\|_G \leq 2 \cdot 10^{-8}$ on G^5 .

Modifications	-	M5	M6	M5 and M6
Work Units	73.62	74.32	56.96	65.57
Execution Time (Seconds)	3.09	3.24	2.58	3.01
$\bar{u}_f$	.731	.738	.669	.704

Table 5.2: Solution of (5.3), (5.4) with $M = 5$, $R = 32/15$
and $\epsilon^M = 2.10^{-8}$ for modifications 5 and 6.
(Run #X35563)

We conclude from the results given in Tables 5.1 and 5.2 that the use of modification 6 yields substantial improvements.

Finally, in Table 5.3 we extend Table 4.2 by comparing the measured execution times for the projected SOR method and the best modification of PFAS (ψ defined by (5.15) and S_k^{k-1} defined by injection) for the dam problem for various values of M . In each case, the iterations were continued until $\|\bar{v}_u\|_G \leq 2 \cdot 10^{-8}$.

M G^M		2 5 × 7	3 9 × 13	4 17 × 25	5 33 × 49	6 65 × 97
G^M Projected SOR	G^M iterations	19	34	69	146	295
	Execution Time (seconds)	.02	.09	.60	4.88	39.37
PFASMD (M6)	G^M work units	23	30.5	38.7	42.8	45.7
	Execution Time (seconds)	.04	.12	.41	1.64	6.57

Table 5.3: Comparison of G^M Projected SOR and PFASMD (modification M6)
for the dam problem with $M = 5$ and $\epsilon^M = 2.10^{-8}$.
(Run #'s X35584 and X35564)

As can be seen from Table 5.3, PFASMD is better than projected SOR except for very small grids.

6. PFMG (PROJECTED FULL MULTIGRID ALGORITHM)

In this section we describe PFMG (Projected Full Multigrid Algorithm) which is a modification of the Full Multigrid Algorithm of Brandt. The flowchart for PFMG is given in Figure 6.1. PFMG has been implemented as a FORTRAN subroutine for the case when Ω is a rectangle in R^2 , and $\mathcal{L}$ is the Laplacian operator. This subroutine is listed in Appendix C as part of the program for solving the porous flow free boundary problem of Section 4, and the problem (5.3), (5.4).

PFMG differs from PFASMD in the following respects.

I. Instead of beginning on G^M , one begins on a coarser grid G^{LIN} and gradually works up to G^M .

The computations begin on the initial grid G^l , $l = LIN$, with an initial approximation $\bar{u}^l$. $\bar{u}^l$ is computed to the required accuracy using grids G^1 through G^l as in the PFASMD implementation of PFAS, except that, as will be discussed below, the decision to move to a different grid is based on slightly different criteria.

Once $\bar{u}^l$ has been found to sufficient accuracy, the initial approximation $\bar{u}^{l+1}$ is obtained from

$$\bar{u}^{l+1} = J_{\ell}^{l+1} \bar{u}^l, \quad (6.1)$$

where J_{ℓ}^{l+1} is an interpolation operator taking grid functions on G^l into grid functions on G^{l+1} . It is known (Brandt [1977, p. 377]) that J_{ℓ}^{l+1} should be more accurate than I_{ℓ}^{l+1} in order to preserve the smoothness of $\bar{u}^l$.

In PFMG J_{ℓ}^{l+1} is implemented as a subroutine INTRP3 which uses cubic interpolation. (To use INTRP3 we must have $l \geq 2$ and so $LIN \geq 2$.) INTRP3 is based upon repeated use of the cubic interpolation formulas

$$f\left(\frac{1}{2}\right) = [-f(-1) + 9f(0) + 9f(1) - f(2)]/16, \quad (6.2)$$

$$f\left(\frac{3}{2}\right) = [f(-1) - 5f(0) + 15f(1) + 5f(2)]/16. \quad (6.3)$$

Repeating this process, we finally obtain an initial approximation $\bar{u}^M$ on G^M . Thereafter, the computation proceeds essentially as in PFASMD.

INPUT PARAMETERS: (Error Control): TOL, RATIO, δ , η , WMAX, WMAXM, PREC, PRECM
 (Iteration Control): M, LIN, NCYCL, NCYCM, NR1, NR2
 (Problem Data): NXO, NYO, HO
 INPUT SUBROUTINES: F, G, SOLRED

Call GROFN to set up storage arrays
 Call SOLRED to read exact solution if available
 WU = 0.

$\ell = \text{LIN}$
 Call PUTF to initialize u^ℓ

Begin new level ℓ
 Call PUTF to initialize $\bar{b}^\ell = h_\ell^2 F^\ell$
 TOLL = TOL * (RATIO** ℓ)
 $\epsilon^\ell = \text{TOLL}$
 WU = .25 WU
 ICYC = 0
 (NCYCL, WMAXL, PRECL) = ((NCYC, WMAX, PREC) if $L < M$
 (NCYCM, WMAXM, PRECM) if $L = M$

$k = \ell$; IR2(ℓ) = 0

Begin new level k
 (5) $\|v_u^k\|_G + 10^{30}$; IR1 = 0

Begin new sweep
 (3) $\|v_u^k\|_G + \|v_u^k\|_G$

SUBROUTINE RELSW
 $\bar{u}^k$ = Projected Gauss-Seidel [$\bar{u}$; A, $\bar{b}$]
 Compute $\|v_u^k\|_G$

WU = WU + 4**($k - \ell$); IR1 = IR1 + 1; IR2(k) = IR2(k) + 1

WU = WMAXL?
 Yes
 No

$\|v_u^k\|_G < \epsilon^k$?
 Yes
 No

IR2(k) = NR2?
 Yes
 No

(8)

IR1 = NR1?
 Yes
 No

$\|v_u^k\|_G < \eta \|v_{old}^k\|_G$?
 Yes
 No

IR1 = 1?
 Yes
 No

$k = k$?
 Yes
 No

II. $\bar{u}^k$ is used to estimate the local truncation error on G^{k-1} .

Suppose that the difference approximations are of order p and that $\bar{u}^k$ can be extended to a smooth function on Ω . Then on G^{k-1} ,

$$A^{k-1}_{I_k} \bar{u}^{k-1-k} \doteq h_{k-1}^2 \bar{u}^k + \tau^{k-1}, \quad (6.4)$$

and

$$S_k^{k-1} A^{k-1}_{I_k} \bar{u}^{k-1-k} \doteq h_k^2 \bar{u}^k + 2^{-(p+2)} \tau^{k-1}, \quad (6.5)$$

where the local truncation error τ^{k-1} depends upon the derivatives of $\bar{u}^k$. Eliminating the unknown $\bar{u}^k$ we obtain

$$\tau^{k-1} \doteq \frac{2^p}{2^p - 1} [A^{k-1}_{I_k} \bar{u}^{k-1-k} - 4S_k^{k-1} A^{k-1}_{I_k} \bar{u}^{k-1-k}], \quad (6.6)$$

$$= \frac{2^p}{2^p - 1} [(4S_k^{k-1} (\bar{b}^k - A^{k-1}_{I_k} \bar{u}^k)) + \{A^{k-1}_{I_k} \bar{u}^{k-1-k} - \{4S_k^{k-1} \bar{b}^k\}]. \quad (6.7)$$

In PFMG, the first { } in (6.7) is evaluated in subroutine RESSW; the second { } is computed and added to the first using subroutines CORSRE and PUTU; the third { } is evaluated in subroutine RESBW (which is a minor modification of RESSW); and, finally, τ^{k-1} is estimated in subroutine TAUCAM. The estimate (6.7) is not accurate near the discrete interface, and so TAUCAM computes τ_z^{k-1} where

$$\tau_z^{k-1}(x) = \begin{cases} \tau^{k-1}(x), & \text{if } \bar{u}^{k-1}(x) > 0 \\ 0, & \text{if } \bar{u}^{k-1}(x) = 0. \end{cases} \quad (6.8)$$

Because of the lack of smoothness of the solution near the free boundary, it is not entirely clear what the value of p should be. It is known (Brezzi and Sacchi [1976]) that the convergence of the finite difference approximations is probably only $O(h^1)$ in the $W^{1,2}(\Omega)$ norm, and Nitsche [1975] has proved $O(h^2 \ln h)$ convergence in the infinity norm. However, these are global error bounds, while we are concerned with the asymptotic behavior of the local truncation error τ . Except in a neighborhood of the discrete interface Γ^ℓ , p is clearly equal to 2. Since the choice of p may vary over Ω , we could perhaps set $p = 1$ near Γ^ℓ , but the values of τ near Γ^ℓ are not very accurate and so, for simplicity, we have taken $p = 2$ everywhere.

III. As usual in numerical analysis the estimate (6.7) for τ^{k-1} can be used in two ways:

(a) To estimate the error $\bar{u}^k - u$.

Since $\tau^k \doteq 2^{-2-p} \tau^{k-1}$, and remembering that G^k has four times as many points as G^{k-1} but $h_{k-1} = 2h_k$, we see from (2.23) that

$$\|\tau_z^k\|_G \doteq \|\tau_z^{k-1}\|_{G/2^p}. \quad (6.9)$$

Combining (6.7), (6.8) and (6.9) we obtain an estimate for $\|\tau_z^k\|_G$.

In the previous sections we were concerned with asymptotic convergence. That is, we were concerned with the rate of convergence of $\bar{u}^k$ to $\bar{u}^k$ over a very large number of iterations. However, if we want an approximation to the solution u of (1.1), it is only necessary to iterate until the residual on G is small compared with the truncation error, that is, until

$$\|\bar{v}_u^k\|_G = O(\|\tau_z^k\|_G). \quad (6.10)$$

Once (6.10) holds, further computation will improve the accuracy of $\bar{u}^k$ as a solution of the finite difference equations but will not improve its accuracy as an approximation to u . Noting (6.9), we see that (6.10) will certainly be true if

$$\|\bar{v}_u^k\|_G \leq \|\tau_z^{k-1}\|_G. \quad (6.11)$$

The stopping criterion (6.11) is incorporated in PFMG by setting

$$\epsilon^l = \max\{\text{PRECL} * \|\tau_z^{l-1}\|_G, \text{TOL} * \text{RATIO} * L\} \quad (6.12)$$

where PRECL, TOL, and RATIO are input parameters. (If $\text{TOL} = 0$, $\text{RATIO} = 1$, and $\text{PRECL} = 1$ then (6.12) reduces to (6.11) for $k = l$).

(b) Improvement of accuracy of $\bar{u}^{k-1}$.

Once an estimate for the truncation error τ_z^{k-1} is available, it can be used to improve the accuracy of the difference approximation on G^{k-1} by replacing $F^{k-1}(x)$ by $F^{k-1}(x) + \tau_z^{k-1}(x)$ (see (6.4)). This is only done at points $x \in G^{k-1}$ such that $\bar{u}^{k-1}(y) > 0$ for all four neighbors $y \in G^{k-1}$ of x since the value of τ_z^{k-1} is not

accurate elsewhere. In PFMG this is done in the subroutine TAUCAM when $k = \ell - 1$ and the input parameter $ITAU = 1$.

Of course, this is only meaningful when $\|\tau_z^{k-1}\|_G$ is small compared to $\|\nabla u^k\|_G$: if the iterations are continued for a long time then convergence will not occur because the conditions of Lemma 5.1 will be violated, but PFMG is never used in this way. In fact, experience with equalities indicates that when τ -extrapolation is used, the best procedure is to avoid relaxation after returning for the last time to the finest grid.

IV. As already mentioned, the logic of PFMG is more complicated than that of PFAS; it is best understood by consulting Figure 6.1 and Appendix C. Several parameters are introduced and this enables one to control explicitly the number of G^k projected sweeps at any level k , and the number of cycles at level ℓ . If

$$\begin{aligned} NR1 &= NR2 = NCYC = NCYCM = -1, \\ PREC &= PRECM = 0, \\ RATIO &= 1, \end{aligned}$$

then the logic of PFMG reduces to that of PFAS.

We now describe numerical results obtained using PFMG to solve the dam problem (4.1), (4.2). In all cases, G^1 is a $(2 + 1) \times (3 + 1)$ grid and $LIN = 2$.

To control the iterations we set $NR1 = 2$, $NR2 = 3$, $NCYC = 1$, $NCYCLN = 3$, and $NCYCM = 10$. The result is that in each cycle on grid G^ℓ , two G^k projected sweeps are carried out for $1 < k \leq \ell$ as we descend from G^ℓ to G^1 , and one G^k projected sweep is carried out as we ascend from G^1 to G^ℓ . For $\ell = LIN$ up to three G^ℓ cycles are allowed, so that a good initial approximation can be obtained. For $LIN < \ell < M$ only one G^ℓ cycle is allowed, while up to 10 G^M cycles are allowed. This will be clearer after consulting Figure 6.2 which shows the output for $M = 4$.

```

..... 2 .....
LEVEL 2 RESIDUAL NORM= .266+001 WORK= 1.000 IR1= 1 IR2(K)= 1
LEVEL 2 RESIDUAL NORM= .174+001 WORK= 2.000 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .920+000 K= 1
LEVEL 1 RESIDUAL NORM= .803+000 WORK= 2.250 IR1= 1 IR2(K)= 1
LEVEL 1 RESIDUAL NORM= .130+000 WORK= 2.500 IR1= 2 IR2(K)= 2
LEVEL 1 RESIDUAL NORM= .814-002 WORK= 2.750 IR1= 3 IR2(K)= 3
LEVEL 2 RESIDUAL NORM= .889+000 WORK= 3.750 IR1= 1 IR2(K)= 3
LEVEL 2 RESIDUAL NORM= .258+000 WORK= 4.750 IR1= 1 IR2(K)= 1
LEVEL 2 RESIDUAL NORM= .102+000 WORK= 5.750 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .924+000 K= 1
LEVEL 1 RESIDUAL NORM= .238-001 WORK= 6.000 IR1= 1 IR2(K)= 1
LEVEL 1 RESIDUAL NORM= .149-002 WORK= 6.250 IR1= 2 IR2(K)= 2
LEVEL 1 RESIDUAL NORM= .930-004 WORK= 6.500 IR1= 3 IR2(K)= 3
LEVEL 2 RESIDUAL NORM= .484-001 WORK= 7.500 IR1= 1 IR2(K)= 3
LEVEL 2 RESIDUAL NORM= .157-001 WORK= 8.500 IR1= 1 IR2(K)= 1
LEVEL 2 RESIDUAL NORM= .443-002 WORK= 9.500 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .983+000 K= 1
LEVEL 1 RESIDUAL NORM= .956-003 WORK= 9.750 IR1= 1 IR2(K)= 1
LEVEL 1 RESIDUAL NORM= .597-004 WORK= 10.000 IR1= 2 IR2(K)= 2
LEVEL 1 RESIDUAL NORM= .373-005 WORK= 10.250 IR1= 3 IR2(K)= 3
LEVEL 2 RESIDUAL NORM= .117-002 WORK= 11.250 IR1= 1 IR2(K)= 3
GREEN NORM OF TAU-Z = .983+000 K= 1
SOLUTION ERROR: L INFINITY NORM = .60769+000 GNORM = .19669+000
SOLUTION : L INFINITY NORM = .28800+003 GNORM = .12949+003
RELATIVE ERROR: L INFINITY NORM = .21100-002 GNORM = .15189-002
..... 3 .....
LEVEL 3 RESIDUAL NORM= .946+000 WORK= 3.812 IR1= 1 IR2(K)= 1
LEVEL 3 RESIDUAL NORM= .265+000 WORK= 4.812 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .114+001 K= 2
LEVEL 2 RESIDUAL NORM= .696-001 WORK= 5.062 IR1= 1 IR2(K)= 1
LEVEL 2 RESIDUAL NORM= .257-001 WORK= 5.313 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .140+001 K= 1
LEVEL 1 RESIDUAL NORM= .138-001 WORK= 5.375 IR1= 1 IR2(K)= 1
LEVEL 1 RESIDUAL NORM= .143-002 WORK= 5.437 IR1= 2 IR2(K)= 2
LEVEL 1 RESIDUAL NORM= .894-004 WORK= 5.500 IR1= 3 IR2(K)= 3
LEVEL 2 RESIDUAL NORM= .115-001 WORK= 5.750 IR1= 1 IR2(K)= 3
LEVEL 3 RESIDUAL NORM= .157+000 WORK= 6.750 IR1= 1 IR2(K)= 3
GREEN NORM OF TAU-Z = .125+001 K= 2
SOLUTION ERROR: L INFINITY NORM = .32441+000 GNORM = .47357+000
SOLUTION : L INFINITY NORM = .28800+003 GNORM = .43262+003
RELATIVE ERROR: L INFINITY NORM = .11264-002 GNORM = .10947-002
..... 4 .....
LEVEL 4 RESIDUAL NORM= .609+000 WORK= 2.687 IR1= 1 IR2(K)= 1
LEVEL 4 RESIDUAL NORM= .189+000 WORK= 3.687 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .646+000 K= 3
LEVEL 3 RESIDUAL NORM= .953-001 WORK= 3.937 IR1= 1 IR2(K)= 1
LEVEL 3 RESIDUAL NORM= .637-001 WORK= 4.187 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .114+001 K= 2
LEVEL 2 RESIDUAL NORM= .351-001 WORK= 4.250 IR1= 1 IR2(K)= 1
LEVEL 2 RESIDUAL NORM= .194-001 WORK= 4.312 IR1= 2 IR2(K)= 2
GREEN NORM OF TAU-Z = .108+001 K= 1
LEVEL 1 RESIDUAL NORM= .913-002 WORK= 4.328 IR1= 1 IR2(K)= 1
LEVEL 1 RESIDUAL NORM= .984-003 WORK= 4.344 IR1= 2 IR2(K)= 2
LEVEL 1 RESIDUAL NORM= .615-004 WORK= 4.359 IR1= 3 IR2(K)= 3
LEVEL 2 RESIDUAL NORM= .942-002 WORK= 4.422 IR1= 1 IR2(K)= 3
LEVEL 3 RESIDUAL NORM= .369-001 WORK= 4.672 IR1= 1 IR2(K)= 3
LEVEL 4 RESIDUAL NORM= .134+000 WORK= 5.672 IR1= 1 IR2(K)= 3
GREEN NORM OF TAU-Z = .860+000 K= 3
SOLUTION ERROR: L INFINITY NORM = .48932-001 GNORM = .23107+000
SOLUTION : L INFINITY NORM = .28800+003 GNORM = .15697+004
RELATIVE ERROR: L INFINITY NORM = .16990-003 GNORM = .14721-003
**** TIME AT ELAPSE IS .1350 SECONDS ****

```

Figure 6.2: Typical output for the PFMG algorithm
(M = 6, Dam Problem, Run #X67705)

Before discussing how the error was controlled, it is necessary to distinguish between the goals of PFMG and PFAS. Asymptotically, PFMG and PFAS behave the same, because once PFMG has reached level M it performs essentially like PFAS. The purpose of PFMG is to obtain quickly an approximation $\bar{u}^M$ which satisfies the stopping criterion (6.11), namely

$$\|\nabla \bar{u}^M\|_G \leq \|\tau_z^{M-1}\|_G.$$

To achieve this we set

PRECM = 1, TOL = 0, ETA = 10 ,

DELTA = 0, PREC = 0, RATIO = 1 .

Finally, we set WMAX = 30, and WMAXM = 40, though these values were of course never reached.

PFMG includes the option of computing, $\|\bar{u}^L - u\|_\infty$ and $\|\bar{u}^L - u\|_G$, where u is the exact solution. For the dam problem, it is possible to compute u analytically using elliptic integrals (Cryer [1976]) but this has not yet been done: we therefore took u to be the most accurate approximation known to us, namely the approximation $\bar{u}^7$ computed in double precision on a $(128 + 1) \times (192 + 1)$ grid as described in Section 4. For problem (5.3), (5.4) the exact solution is given by (5.4).

We first performed a number of experiments with $M = 2, 3, 4$, and 5:

1. τ -extrapolation (with $p = 2$) gave slightly worse results for the dam problem and problem (5.3), (5.4).

2. In contrast to our experience with PFAS, the use of modification 6 had only a slight effect.

3. It was thought that convergence might be improved by multiplying the difference $\nabla \bar{u}^k(x)$ by h for points x near the free boundary before computing $\|\nabla \bar{u}^k(x)\|_G$. This was implemented as a subroutine RELAX1 but was found to have negligible effect.

All the results given below are for the case of no τ -extrapolation (ITAU = 0) and no modification (NINTSW = NRESSW = 1).

The results for the dam problem for different values of M are shown in Table 6.1.

M	2	3	4	5
G^M Work Units	3.75	6.75	5.67	5.41
Execution Time (seconds)	.009	.053	.131	.349
$\ \tilde{u}^M - \tilde{u}^7\ _\infty / \ u\ _\infty$	.00374	.00112	.000169	.0000623
$\ \tilde{u}^M - \tilde{u}^7\ _G / \ u\ _G$	.00334	.00109	.000147	.0000405
$\ \nabla \tilde{u}^M\ _G$	.889	.157	.134	.0714
$\ \tau_2^{M-1}\ _G$	2.39	1.25	0.86	0.60

Table 6.1: Solution of the dam problem using PFMG.

(Run #X67247)

Since we only have estimates for τ^{M-1} , it is not possible to obtain rigorous error bounds. Nevertheless, it is interesting to apply the error bounds of Section 2.

Let $\tilde{u}^M$ denote the vector obtained by evaluating the solution $u(x)$ on G^M . Then, from (6.4), (1.1), (2.2), (2.3), (2.13), and (3.1),

$$A^M \tilde{u}^M \leq b^M + \tau^M,$$

so that, from Lemma 2.1,

$$\|\tilde{u}^M - u^M\|_2 \leq \frac{1}{\alpha_M} \|\tau_+^M\|_2. \quad (6.13)$$

On the other hand, from Lemma 2.2,

$$\|u^M - \tilde{u}^M\|_2 \leq \frac{1}{\alpha_M} \|P^M\|_2 \|\nabla \tilde{u}^M\|_2.$$

For the dam problem, P is an upper triangular matrix with at most two nonzero elements per row, and $\|P^M\|_2 \leq 2$. Thus,

$$\|u^M - \tilde{u}^M\|_2 \leq \frac{2}{\alpha_M} \|\nabla \tilde{u}^M\|_2. \quad (6.14)$$

Combining these inequalities we obtain

$$\|\tilde{u}^M - \bar{u}^M\|_2 \leq \frac{1}{\alpha_M} [\|\tau_+^M\|_2 + 2\|\nabla \bar{u}^M\|_2] ,$$

or, equivalently,

$$\|\tilde{u}^M - \bar{u}^M\|_G \leq \frac{1}{\alpha_M} [\|\tau_+^M\|_G + 2\|\nabla \bar{u}^M\|_G] . \quad (6.15)$$

Using (6.8) and (6.9), we conclude that

$$\|\tilde{u}^M - \bar{u}^M\|_G \leq \frac{1}{\alpha_M} \left[\frac{1}{2^p} \|\tau_z^{M-1}\|_G + 2\|\nabla \bar{u}^M\|_G \right] . \quad (6.16)$$

Next, we note that for the dam problem

$$\alpha_M \doteq \alpha h_M^2 , \quad (6.17)$$

where

$$\alpha = \left(\frac{\pi}{16}\right)^2 + \left(\frac{\pi}{24}\right)^2 \doteq .055 > 14/256 . \quad (6.18)$$

and

$$h_M = 16 \cdot 2^{-M}$$

Thus, finally, for the dam problem,

$$\|\tilde{u}^M - \bar{u}^M\|_G \leq \frac{2^{2M}}{14} \left[\frac{1}{2^p} \|\tau_z^{M-1}\|_G + 2\|\nabla \bar{u}^M\|_G \right] . \quad (6.19)$$

For example, for $M = 5$ we obtain, using Table 6.1, that

$$\begin{aligned} \|\tilde{u}^5 - \bar{u}^5\|_G / \|\tilde{u}^5\|_G &\leq \frac{2^{10}}{14} \left[\frac{1}{4} (0.60 + 2(.071)) / (5.9 \cdot 10^4) \right] \\ &\doteq .00036 ; \end{aligned} \quad (6.20)$$

the observed value quoted in Table 6.1 is .000040.

In Table 6.2 we repeat the computations of Table 6.1 for the problem (5.3), (5.4).

M	2	3	4	5
G^M Work Units	3.75	6.75	5.672	5.414
Execution Time (seconds)	.028	.103	.263	.842
$\ \bar{u}^M - \bar{u}^M \ _\infty / \ u \ _\infty$	.0147	.000985	.000266	.0000645
$\ \bar{u}^M - \bar{u}^M \ _G / \ \bar{u}^M \ _G$	.0147	.00127	.000376	.0000956
$\ \bar{u}^M \ _G$	10.5	.241	.121	.0764
$\ \tau_z^{M-1} \ _G$	4.18	1.62	1.10	.749

Table 6.2 Solution of problem (5.3), (5.4) using PFMG.
(Run #X67243)

The error estimate (6.19) also holds for the problem (5.3), (5.4), since we are using the Laplace operator on a rectangle with sides in the ratio 2:3. Applying (6.19) we obtain

$$\| \bar{u}^5 - \bar{u}^5 \|_G / \| \bar{u}^5 \|_G \leq \frac{2^{10}}{14} \left| \frac{1}{4} (.75) + 2 (.076) \right| / (1.2 \cdot 10^4),$$

$$\doteq .0021 ;$$

the observed value quoted in Table 6.2 is .0000645.

The behavior of the global error $\bar{u}^M - u$ can be checked using Tables 6.1 and 6.2. From Table 6.1 we have

$$\left[\frac{\| \bar{u}^5 - \bar{u}^7 \|_\infty}{\| \bar{u}^2 - \bar{u}^7 \|_\infty} \right]^{1/3} = \left[\frac{.0000623}{.00374} \right]^{1/3} \doteq \frac{1}{2^{1.96}}$$

In Table 6.2 the error in $\bar{u}^2$ is "abnormally large". However,

$$\left[\frac{\| \bar{u}^5 - u \|_\infty}{\| \bar{u}^3 - u \|_\infty} \right]^{1/2} = \left[\frac{.0000645}{.000985} \right]^{1/2} \doteq \frac{1}{2^{1.96}}$$

These results strongly suggest that the global error is $O(h^2)$.

The behavior of the local error τ can also be checked using Tables 6.1 and 6.2. From Table 6.1,

$$[\|\tau_z^4\|_G / \|\tau_z^1\|_G]^{1/3} = [.60/2.39]^{1/3} \doteq 1/2^{.66},$$

while, from Table 6.2,

$$[\|\tau_z^4\|_G / \|\tau_z^1\|_G]^{1/3} = [4.18/.749]^{1/3} \doteq 1/2^{.82},$$

so that $\tau = O(h^q)$ with $q \in (.66, .82)$. This explains why τ -extrapolation with $p = 2$ did not reduce the computational effort for these problems. The essential difficulty is of course that the irregularity of the discrete interface makes it difficult to obtain accurate estimates for τ . In fact, τ -extrapolation with $p = 1$ was found to be slower than τ -extrapolation with $p = 2$.

Finally, in Table 6.3 we repeat the computations of Table 5.3 for a tolerance $\epsilon^M = .0714$, the value of $\|\nabla u^5\|_G$ in Table 6.1. We are thus comparing the performance of PFAS (with modification 6), PFMG, and projected SOR for comparable errors.

Method	PFMG	PFASMD (M6)	Projected SOR
Work Units	5.41	9.64	56.0
$\ \nabla u^M\ _G$	.0714	.0239	.0695
Execution Time (seconds)	.349	.440	1.94

Table 6.3: Solution of the dam problem for $M=5$ and $\epsilon^M = .0714$
using PFASMD (modification 6), PFMG, and projected SOR.
(Runs #X67247 and #X67250)

From Table 6.3, we see that PFMG is faster than projected SOR even when only low accuracy is required. PFAS and PFMG require comparable times, but PFMG gives much more information and is, therefore, preferable. PFMG also uses fewer work units than PFAS. This is significant because the number of work units used is independent of

the computer. Furthermore, on the basis of experience with many problems, it can be said that the number of work units used does not vary greatly with the problem: for most operators $\mathcal{L}$ PFMG requires only 5.4 work units.

We conclude this section with some remarks on the implementation of PFMG:

1. From Table 6.3 we see that the execution time per work unit of PFMG is greater than the comparable quantity for PFAS by a factor

$$\frac{.349}{5.41} / \frac{.440}{9.64} = 1.41 .$$

This additional overhead is probably due to the cubic interpolation used by J_{k-1}^k , and could perhaps be reduced by better programming. When $\mathcal{L}$ is complicated, the additional overhead required by PFMG is relatively much less significant: it is only with a very simple operator like the 5-point Laplacian that the additional overhead is so expensive.

2. In PFMG one often need not have any storage for the finest grid G^M - not even external storage. The algorithm visits G^M only twice: at the beginning of the last cycle and at the end of the last cycle.

At the beginning of the cycle, the following operations are performed: interpolation (J_{M-1}^M); two G^M projected sweeps; and residual transfer (I_M^{M-1} and S_M^{M-1}). All these operations can be made in one passage over G^M , in such a way that only four columns of G^M are held in memory at one time. Each time a new column, say column i , is created (by interpolation), a relaxation can be made in column $i-1$, then the second relaxation can already be made in column $i-2$ and the residuals from column $i-3$ can be transferred back to the coarse grid. Column $i-4$ can simultaneously be discarded (i.e., replaced by column i). After this visit to G^M all the information is available (in $\bar{F}^{M-1}$ and $\bar{u}^{M-1}$) to solve the G^{M-1} problem to the truncation level of G^M .

The final return to G^M (which would require the storage of the previous values of U^M) is made in order to obtain the solution on G^M rather than on G^{M-1} , but it does not improve its pointwise accuracy. If one is only interested in knowing some functionals of the solution, these can be calculated without having the final solution on G^M . To approximate a functional $\mathcal{K}(U)$, for example, one computes $\mathcal{K}(\bar{u}^{M-1}) + \epsilon_M^{M-1}$,

where $\sigma_M^{M-1} = \mathcal{H}(\bar{u}^M) - \mathcal{H}(I_M^{M-1-M} \bar{u}^M)$, $\bar{u}^{M-1}$ is the final solution on G^{M-1} , and $\bar{u}^M$ is the last solution on G^M before switching back to G^{M-1} . Clearly, σ_M^{M-1} can be calculated during the above-mentioned passage on G^M . Note that σ_M^{M-1} is a "relative truncation correction", similar to τ_M^{M-1} . It makes the approximation $\mathcal{H}(\bar{u}^{M-1}) + \sigma_M^{M-1}$ correct to the G^M truncation level. $\mathcal{H}$ need not be a linear functional.

7. CONCLUSIONS AND RECOMMENDATIONS.

1. Multigrid methods can easily be adapted to handle linear complementarity problems arising from free boundary problems (see Table 4.2).
2. Multigrid methods are superior to projected SOR and modified block SOR (see Tables 5.3 and 6.3, and Section 4).
3. For high accuracy solutions of the discrete LCP, one should use PFASMD with modification 6 (see Tables 5.1 and 5.2).
4. For solutions which are accurate to within truncation error one should use PFMG, with no modifications (see Tables 6.1, 6.2, and 6.3).

Finally, we conclude with some comments suggesting possible future applications of multigrid methods to complementarity problems:

1. For equalities, experience has shown that multigrid methods are as efficient for problems where f is nonlinear as for problems where f is linear.
2. Experience from equalities indicates that with similar efficiency (just a few more work units) one can solve much more difficult problems, such as problems in which the coefficients of f vary by orders of magnitude (e.g., large variations in the diffusivity of the dam). In such cases SOR and other methods converge very slowly. See Alcouffe et. al. (to appear).
3. The truncation error near a discrete interface cannot be reduced by using higher order approximations because the second derivatives are usually discontinuous. A good way to improve the approximation would be to use finer mesh sizes near the discrete interface. This can be combined very effectively with the multigrid process (see Brandt [1979, Section 3]). In fact, a vast improvement is expected if r -extrapolation is used together with local refinements. Fine levels will then be used only near the interface.

Acknowledgement.

We thank Professor R. Sacher for making available a copy of his program for solving LCP's using the modified block SOR algorithm of Cottle and Sacher, and for his comments on an early version of this report.

AB/CWC/ed

REFERENCES

- R. ALCOUFFE, A. BRANDT, J. DENDY and J. PAINTER. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. Los Alamos Scientific Laboratory Report, to appear.
- C. BAIOCCHI. Sur un probleme a frontiere libre traduisant le filtrage de liquides a travers des milieux poreux. Comptes Rendus Acad. Sci. Paris, A273(1971), pp. 1215-1217.
- C. BAIOCCHI. Free boundary problems and variational inequalities. Technical Summary Report No. 1883, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1978.
- M. L. BALINSKI and R. W. COTTLE. Complementarity and Fixed Point Problems. North-Holland, Amsterdam, 1978.
- J. BEAR. Dynamics of Fluids in Porous Media. American Elsevier, New York, 1972.
- A. BRANDT. Multi-level adaptive solutions to boundary value problems. Math. Computation, 31(1977), pp. 333-390.
- A. BRANDT. Multi-level adaptive techniques (MLAT) for singular-perturbation problems. In Numerical Analysis of Singular Perturbation Problems, P. W. Hemker and J. J. H. Miller (editors). New York, Academic Press 1979, pp. 53-142.
- A. BRANDT and N. DINAR. Multi-grid solutions to elliptic flow problems. Symposium on Numerical Solution of Partial Differential Equations, S. V. Parter, ed. New York, Academic Press, 1979, pp. 53-147.
- F. BREZZI and G. SACCHI. A finite element approximation of variational inequalities related to hydraulics. Calcolo, 13(1976), pp. 259-273.
- J. CEA, R. GLOWINSKI, and J. C. NEDELEC. Application des methodes d'optimisation, de differences et d'elements finis a l'analyse numerique de la torsion elasto-plastique d'une barre cylindrique. In Approximation et Methodes Iteratives de Resolution d'Inequations Variationnelles et de Problemes Non Lineaires. Cahier de l'IRIA, No. 12, 1974, pp. 7-138.
- G. CIMATTI. On a problem of the theory of lubrication governed by a variational inequality. Applied Math. and Optimization, 3(1977), pp. 227-242.
- R. W. COTTLE. Computational experience with large-scale linear complementarity problems. Technical Report No. SOL 74-13, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1974.
- R. W. COTTLE, F. GIANNESI, and J. L. LIONS (editors). Variational Inequalities and Complementarity Problems. John Wiley, New York, 1980.
- R. W. COTTLE, G. H. GOLUB, and R. S. SACHER. On the solution of large, structured linear complementarity problems: the block partitioned case. Applied Mathematics and Optimization 4(1978), pp. 347-363.

- R. W. COTTLE and R. S. SACHER. On the solution of large, structured linear complementarity problems: the tridiagonal case. *Applied Mathematics and Optimization*, 3(1977), pp. 321-340.
- C. W. CRYER. The solution of a quadratic programming problem using systematic overrelaxation. *SIAM J. Control*, 9(1971), pp. 385-392.
- C. W. CRYER. The method of Christopherson for solving free boundary problems for infinite journal bearings by means of finite differences. *Math. Comp.*, 25(1971a), pp. 435-444.
- C. W. CRYER. A survey of steady state porous flow free boundary problems. Technical Summary Report No. 1657, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1976.
- C. W. CRYER. A bibliography of free boundary problems. Technical Summary Report No. 1793, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1977.
- C. W. CRYER. The solution of the axisymmetric elastic-plastic torsion of a shaft using variational inequalities. Technical Summary Report No. 1948, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1979. To appear in *J. Math. Anal. Appl.*
- C. W. Cryer. Successive overrelaxation methods for solving linear complementarity problems arising from free boundary problems. *Proceedings, Seminar on Free Boundary Problems, Pavia, October 1979a*, to appear.
- G. DUVAUT and J. L. LIONS. *Inequalities in Mechanics and Physics*. Dunod, Paris, 1976.
- R. S. FALK. Error estimates for the approximation of a class of variational inequalities. *Math. Computation*, 28(1974), pp. 963-971.
- R. GLOWINSKI. La methode de relaxation. *Rendiconti di Matematica*, 14 (1971), 56 pages.
- R. GLOWINSKI. Finite elements and variational inequalities. Technical Summary Report No. 1885, Mathematics Research Center, University of Wisconsin, 1978.
- R. GLOWINSKI, J. L. LIONS, and R. TREMOLIERES. *Analyse Numerique des Inequations Variationnelles*. Dunod, Paris, 1976.
- D. KINDERLEHRER and G. STAMPACCHIA. *An Introduction to Variational Inequalities and their Applications*. Academic Press, New York, 1980.
- H. LANCHON. Torsion elastoplastique d'un arbre cylindrique de section simplement ou multiplement connexe. *J. Mecanique*, 13(1974), pp. 267-320.
- J. A. NITSCHKE. L-infinity convergence of finite element approximations. In *Mathematical Aspects of the Finite Element Method*, Rome Italy, 1975.
- R. S. Varga. *Matrix Iterative Analysis*. Prentice Hall, Englewood Cliffs, N. J., 1962.

===== APPX-A-PFAS =====

```

1.      C      *****
2.      C
3.      C      THIS PROGRAM SOLVES THE PROBLEM OF POROUS FLOW THROUGH A
4.      C      RECTANGULAR DAM OF HEIGHT Y1 AND WIDTH A.
5.      C      THE RESERVOIR TO THE RIGHT OF THE DAM IS OF HEIGHT Y2.
6.      C
7.      C      WRITTEN BY ACHI BRANDT AND COLIN CRYER AUGUST 1980
8.      C
9.      C      ADDITIONAL PARAMETERS USED ARE:
10.     C      NX0      THE NUMBER OF GRID INTERVALS IN THE X-DIRECTION IN
11.     C      THE COARSEST GRID, GRID 1.
12.     C      NY0      THE NUMBER OF GRID INTERVALS IN THE Y-DIRECTION IN
13.     C      THE COARSEST GRID, GRID 1.
14.     C      H0       THE GRID SIZE IN THE COARSEST GRID, GRID 1.
15.     C      M        THE NUMBER OF GRIDS TO BE USED.
16.     C      TOL      THE TOLERANCE. COMPUTATION TERMINATES IF THE RESIDUAL
17.     C      ON THE FINEST GRID IS LESS THAN TOL.
18.     C      WMAX     THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
19.     C      FINEST GRID. COMPUTATION TERMINATES WHEN WMAX IS EXCEEDED.
20.     C      IN PRACTICAL CASES, ONE SETS WMAX=30. IN THE PRESENT WORK,
21.     C      WE OFTEN SET WMAX=100 SO AS TO OBSERVE THE ASYMPTOTIC
22.     C      BEHAVIOR OF THE ALGORITHM.
23.     C      MPRINT   THE GRID TO BE PRINTED AT THE END OF THE COMPUTATION.
24.     C      THAT IS, WE PRINT THE MPRINT SUBSET OF THE FINAL ANSWER
25.     C      ON THE GRID M.
26.     C      NQSIZE    SIZE OF ARRAY Q
27.     C      MUST BE CHANGED FOR LARGE PROBLEMS BY EDITING PROGRAM
28.     C      =18000 FOR DAM PROBLEM M=2,3,4,5,6
29.     C      =70000 FOR DAM PROBLEM M=7
30.     C
31.     C
32.     C      ALL THE PARAMETERS ARE SET IN THE PROGRAM, BUT THEIR VALUES
33.     C      CAN BE RESET ON THE NAMELIST INPUT CARD WHICH IS READ IN
34.     C      BY THE PROGRAM.
35.     C      THE NAMELIST CARD MUST BE PROVIDED AS INPUT.
36.     C
37.     C      THE PROGRAM SETS UP STORAGE FOR THE SOLUTIONS AND RIGHT
38.     C      HAND SIDES.
39.     C      THE SOLUTIONS ARE STORED IN ARRAYS 1 TO M.
40.     C      THE RIGHT HAND SIDES ( OR, SOMETIMES THE RESIDUALS )
41.     C      ARE STORED IN ARRAYS M+1 TO 2*M.
42.     C
43.     C      THIS PROGRAM WAS USED TO COMPUTE THE RESULTS IN FIGURE 3.2
44.     C      AND TABLES 4.1 AND 4.2 OF THE MRC REPORT.
45.     C
46.     C      *****
47.     C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
48.     C      EXTERNAL G,F
49.     C      COMMON /PRBDAT/Y1,Y2,A
50.     C      COMMON /QDAT/NQSIZE,NQERR
51.     C      NAMELIST /INDAT/Y1,Y2,A,NX0,NY0,H0,M,TOL,WMAX,MPRINT
52.     C      NQSIZE=18000
53.     C      Y1=24
54.     C      Y2=4
55.     C      A=16
56.     C      NX0=4
57.     C      NY0=6

```

===== APP-A-PFAS =====

```

58.      H0=4.
59.      M=3
60.      TOL=0.
61.      WMAX=30.
62.      MPRINT=1
63.      READ(5,INDAT)
64.      WRITE(6,INDAT)
65.      C      SET TIME TO ZERO
66.      CALL URTIMS(0.0)
67.      CALL PFAS(NX0,NY0,H0,M,TOL,WMAX,G,F)
68.      C      PRINT ELAPSED TIME
69.      T=URTIMG('ELAPSED TIME')
70.      CALL SOLPRT(M,MPRINT)
71.      STOP
72.      END
73.      C
74.      C
75.      DOUBLE PRECISION FUNCTION F(X,Y)
76.      C      DAM PROBLEM
77.      C      THIS SUBROUTINE COMPUTES THE RIGHT HAND SIDE OF THE
80.      C      GOVERNING POISSON EQUATION DEL*DEL U=F.
79.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
80.      F=1.
81.      RETURN
82.      END
83.      C
84.      C
85.      DOUBLE PRECISION FUNCTION G(X,Y)
86.      C      DAM PROBLEM
87.      C      THIS SUBROUTINE COMPUTES THE BOUNDARY DATA AND THE
88.      C      INITIAL APPROXIMATION TO THE SOLUTION U.
89.      C      THE INITIAL APPROXIMATION IS OBTAINED BY LINEAR INTERPOLATION
90.      C      IN THE X-DIRECTION BETWEEN THE GIVEN BOUNDARY DATA.
91.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
92.      COMMON /PRBDAT/Y1,Y2,A
93.      G1=.5*(Y1-Y)**2
94.      G2=.5*(Y2-Y)**2
95.      IF( Y.GE.Y2) G2=0
96.      G=(G1*(A-X)+ G2*X)/A
97.      RETURN
98.      END
99.      C
100.     C
101.     SUBROUTINE PFAS(NX0,NY0,H0,M,TOL,WMAX,U1,F)
102.     C      THIS SUBROUTINE IS THE MAIN MULTIGRID SUBROUTINE.
103.     C      IT INITIALIZES THE PROBLEM, AND REPEATEDLY CALLS
104.     C      THE SUBROUTINES RELAX,RESCAL,PUTU,CORSRE,SUBTRC,AND INTADD.
105.     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
106.     COMMON /QDAT/NQSIZE,NQERR
107.     EXTERNAL U1,F
108.     DIMENSION EPS(10)
109.     C
110.     C
111.     C      SET UP ARRAYS 1 TO M FOR THE SOLUTIONS
112.     C      AND ARRAYS M+1 TO 2*M FOR THE RIGHT HAND SIDES,
113.     C      AND CHECK THAT Q ARRAY IS LARGE ENOUGH
114.     NQERR=0

```

===== APPX-A-PFAS =====

```

115.      DO 1 K=1,M
116.      K2=2**(K-1)
117.      CALL GRDFN(K,NX0*K2+1,NY0*K2+1,H0/K2)
118.      1 CALL GRDFN(K+M,NX0*K2+1,NY0*K2+1,H0/K2)
119.      PRINT 10,NQSIZE
120.      10  FORMAT(' SIZE OF Q ARRAY = ', I10)
121.      IF(NQERR.EQ.0)GOTO 12
122.      PRINT 11,NQERR
123.      11  FORMAT(' *** ERROR IN GRDFN *** ARRAY Q NOT LARGE ENOUGH ***',
124.      * /,' ARRAY Q SIZE SHOULD BE AT LEAST =', I10)
125.      STOP
126.      12  CONTINUE
127.      C
128.      C
129.      C      INITIALIZE
130.      EPS(M)=TOL
131.      K=M
132.      WU=0
133.      CALL PUTF(M,U1,0)
134.      CALL PUTF(2*M,F,2)
135.      ETA=.5
136.      DELTA=.15
137.      C
138.      C      START OF MAIN LOOP IN WHICH ONE MODIFIED GAUSS-SEIDEL
139.      C      SWEEP ON GRID K IS MADE.
140.      C
141.      5  ERR=1.E30
142.      3  ERRP=ERR
143.      CALL RELAX(K,K+M,ERR)
144.      IF (WU .LE. 0) ERRBEG=ERR
145.      WU=WU+4.**(K-M)
146.      WRITE(6,4)K,ERR,WU
147.      4  FORMAT(' LEVEL',I2,' RESIDUAL NORM=', D10.3,' WORK=', F7.3)
148.      IF(ERR.LT.EPS(K))GOTO 2
149.      IF (WU.GE.WMAX)RETURN
150.      IF(K.EQ.1.OR.ERR/ERRP.LT. ETA)GO TO 3
151.      C
152.      C      GO TO COARSER GRID
153.      IF( K.NE.M .OR. WU.LE.3 ) GOTO 92
154.      FMU=0.0
155.      IF( ERR.GT.0 ) FMU=(ERR/ERRBEG)**(1.D0/(WU-1))
156.      PRINT 91,FMU
157.      91  FORMAT(' ', 20('*'),'END OF CYCLE',20('*'),'MU = ',F8.4)
158.      92  CONTINUE
159.      CALL RESCAL(K,K+M,K+M-1)
160.      EPS(K-1)=DELTA*ERR
161.      K=K-1
162.      CALL PUTU(K+1,K)
163.      CALL CORSRE(K,K+M)
164.      GOTO 5
165.      C
166.      C      GO TO FINER GRID
167.      2  IF (K.EQ.M)RETURN
168.      CALL SUBTRC(K+1,K)
169.      CALL INTADD(K,K+1)
170.      K=K+1
171.      GOTO 5

```

===== APPX-A-PFAS =====

```

172.      END
173.      C
174.      C
175.      SUBROUTINE CORSRE(K,KRHS)
176.      C      APPLIES THE DIFFERENCE OPERATOR ON GRID K
177.      C      TO THE GRID FUNCTION IN ARRAY K, AND ADDS THE RESULT TO THE
178.      C      VALUES IN ARRAY KRHS.
179.      C      KRHS      KRHS      K K,0
180.      C      B      = R      + A U
181.      C
182.      C      THE RESULT IS STORED IN ARRAY KRHS.
183.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
184.      C      COMMON Q(18000),IST(200),IRHS(200)
185.      C      CALL KEY(K,IST,II,JJ,H)
186.      C      CALL KEY(KRHS,IRHS,II,JJ,H)
187.      C      I1=II-1
188.      C      J1=JJ-1
189.      C      DO 1 I=2,I1
190.      C      IR=IRHS(I)
191.      C      IO=IST(I)
192.      C      IM=IST(I-1)
193.      C      IP=IST(I+1)
194.      C      DO 1 J=2,J1
195.      C      A=-Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
196.      C      1 Q(IR+J)=-A-4.*Q(IO+J)
197.      C      RETURN
198.      C      END
199.      C
200.      C
201.      SUBROUTINE GRDFN(N,IMAX,JMAX,HH)
202.      C      SETS UP ARRAY N.
203.      C      IMAX      THE DIMENSION IN THE X DIRECTION
204.      C      JMAX      THE DIMENSION IN THE Y DIRECTION
205.      C      HH      THE GRID SIZE
206.      C      THE ARRAY NST CONTAINS THE STARTING ADDRESSES OF THE ARRAYS.
207.      C      THE ARRAY IMX CONTAINS THE MAXIMUM ROW NUMBERS
208.      C      THE ARRAY JMX CONTAINS THE MAXIMUM COL NUMBERS
209.      C      THE ARRAY H      CONTAINS THE GRID SIZES.
210.      C
211.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
212.      C      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
213.      C      COMMON /QDAT/NQSIZE,NQERR
214.      C      DATA IQ/1/
215.      C      NST(N)=IQ
216.      C      IMX(N)=IMAX
217.      C      JMX(N)=JMAX
218.      C      H(N)=HH
219.      C      IQ=IQ+IMAX*JMAX
220.      C      IF(IQ.LE.NQSIZE+1) RETURN
221.      C      NQERR=IQ-1
222.      C      END
223.      C
224.      C
225.      SUBROUTINE INTADD(KC,KF)
226.      C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
227.      C      AND ADDS TO SOLUTION ON GRID KF.
228.      C      KF      KF      KC      KF      KF

```

===== APPX-A-PFAS =====

```

229.      C      U  = PHI( I  W  + U  , U  )
230.      C              KC
231.      C
232.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
233.      COMMON Q(18000),ISTC(200),ISTF(200)
234.      CALL KEY(KC,ISTC,IIC,JJC,HC)
235.      CALL KEY(KF,ISTF,IIF,JJF,HF)
236.      DO 1 IC=2,IIC
237.      IF=2*IC-1
238.      JF=1
239.      IFO=ISTF(IF)
240.      IFM=ISTF(IF-1)
241.      ICO=ISTC(IC)
242.      ICM=ISTC(IC-1)
243.      DO 1 JC=2,JJC
244.      JF=JF+2
245.      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
246.      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
247.      Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
248.      Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
249.      Q(IFO+JF-1)=Q(IFO+JF-1)+A
250.      1 Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
251.      RETURN
252.      END
253.      C
254.      C
255.      SUBROUTINE KEY(K,IST,IMAX,JMAX,HH)
256.      C      RECOVERS THE INFORMATION ABOUT ARRAY K SET UP BY
257.      C      THE SUBROUTINE GRDFN.
258.      C      THE VALUE OF THE GRID FUNCTION AT THE POINT (I,J)
259.      C      IS ADDRESSED AS U(IST(J)+I).
260.      C
261.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
262.      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
263.      DIMENSION IST(1)
264.      IMAX=IMX(K)
265.      JMAX=JMX(K)
266.      IS=NST(K)-JMAX-1
267.      DO 1 I=1,IMAX
268.      IS=IS + JMAX
269.      1 IST(I)=IS
270.      HH=H(K)
271.      RETURN
272.      END
273.      C
274.      C
275.      SUBROUTINE PUTF(K,F,NH)
276.      C      INSERTS THE VALUES OF THE FUNCTION F
277.      C      EVALUATED AT THE POINTS OF GRID K
278.      C      AND MULTIPLIED BY GRIDSIZE**NH
279.      C      INTO THE ARRAY K.
280.      C
281.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
282.      COMMON Q(18000),IST(600)
283.      CALL KEY (K,IST,II,JJ,H)
284.      H2=H**NH
285.      DO 1 I=1,II

```

===== APPX-A-PFAS =====

```

286.      DO 1 J=1,JJ
287.      X=(I-1)*H
288.      Y=(J-1)*H
289.      1 Q(IST(I)+J)=F(X,Y)*H2
290.      RETURN
291.      END
292.      C
293.      C
294.      SUBROUTINE PUTU(KF,KC)
295.      C      THIS SUBROUTINE INJECTS THE SOLUTION ON THE FINE GRID
296.      C      KF INTO THE COARSE GRID KC.
297.      C      KC,0      KC  KF
298.      C      U      = I      U
299.      C      KC
300.      C
301.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
302.      COMMON Q(18000),IUF(200),IUC(200)
303.      CALL KEY(KF,IUF,IIF,JJF,HF)
304.      CALL KEY(KC,IUC,IIC,JJC,HC)
305.      DO 1 IC=1,IIC
306.      IF=2*IC-1
307.      IFO=IUF(IF)
308.      ICO=IUC(IC)
309.      JF=-1
310.      DO 1 JC=1,JJC
311.      JF=JF+2
312.      Q(ICO+JC)=      Q(IFO+JF)
313.      1 CONTINUE
314.      RETURN
315.      END
316.      C
317.      C
318.      SUBROUTINE RELAX(K,KRHS,ERR)
319.      C      CARRIES OUT ONE MODIFIED GAUSS-SEIDEL
320.      C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
321.      C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
322.      C
323.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
324.      COMMON Q(18000),IST(200),IRHS(200)
325.      CALL KEY(K,IST,II,JJ,H)
326.      CALL KEY(KRHS,IRHS,II,JJ,H)
327.      I1=II-1
328.      J1=JJ-1
329.      ERR=0.
330.      DO 1 I=2,I1
331.      IR=IRHS(I)
332.      IO=IST(I)
333.      IM=IST(I-1)
334.      IP=IST(I+1)
335.      DO 1 J=2,J1
336.      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
337.      QT=-.25*A
338.      QN=MAX(0.0,QT)
339.      ERR=ERR+(QN-Q(IO+J))**2
340.      1 Q(IO+J)=QN
341.      ERR=SQRT(ERR)/H
342.      RETURN

```

===== APPX-A-PFAS =====

```

343.      END
344.      C
345.      C
346.      SUBROUTINE RESCAL(KF,KRF,KRC)
347.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
348.      C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
349.      C      BEFORE INJECTION, THE RESIDUAL IS SCALED
350.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
351.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
352.      C      GRIDSIZE ON GRID KC.
353.      C      KRC      KC      KRF      KF      KF
354.      C      R      = 4*S      ( B      - A      U      )
355.      C      KF
356.      C
357.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
358.      C      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
359.      C      CALL KEY(KF,IUF,IIF,JJF,HF)
360.      C      CALL KEY(KRF,IRF,IIF,JJF,HF)
361.      C      CALL KEY(KRC,IRC,IIC,JJC,HC)
362.      C      IIC1=IIC-1
363.      C      JJC1=JJC-1
364.      C      DO 1 IC=2,IIC1
365.      C      ICR=IRC(IC)
366.      C      IF=2*IC-1
367.      C      JF=1
368.      C      IFR=IRF(IF)
369.      C      IFO=IUF(IF)
370.      C      IFM=IUF(IF-1)
371.      C      IFP=IUF(IF+1)
372.      C      DO 1 JC=2,JJC1
373.      C      JF=JF+2
374.      C      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
375.      C      1 Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
376.      C      RETURN
377.      C      END
378.      C
379.      C
380.      C      SUBROUTINE SOLPRT(M,MPRINT)
381.      C      PRINTS THE ARRAY M ON THE SUBARRAY MPRINT.
382.      C
383.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
384.      C      COMMON Q(18000),IST(600)
385.      C      DIMENSION QTEM(100)
386.      C      CALL KEY (M,IST,II,JJ,H)
387.      C      INTERV=2** (M-MPRINT)
388.      C      DO 20 J=JJ,1,-INTERV
389.      C      L=0
390.      C      DO 10 I=1,II,INTERV
391.      C      X AND Y ARE NOT PRINTED HERE, BUT ARE COMPUTED IN
392.      C      CASE A LATER VERSION NEEDS THEM.
393.      C      X=(I-1)*H
394.      C      Y=(J-1)*H
395.      C      L=L+1
396.      C      QTEM(L)=Q(IST(I)+J)
397.      C      10 CONTINUE
398.      C      PRINT *,(QTEM(LL),LL=1,L)
399.      C      20 CONTINUE

```

***** APPX-A-PFAS *****

```

400.      RETURN
401.      END
402.      C
403.      C
404.      SUBROUTINE SUBTRC(KF,KC)
405.      C      THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
406.      C      GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
407.      C      KC      KC      KC      KF
408.      C      W      = U      - I      U
409.      C                      KF
410.      C
411.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
412.      C      COMMON Q(18000),IUF(200),IUC(200)
413.      C      CALL KEY(KF,IUF,IIF,JJF,HF)
414.      C      CALL KEY(KC,IUC,IIC,JJC,HC)
415.      C      DO 1 IC=1,IIC
416.      C      IF=2*IC-1
417.      C      IFO=IUF(IF)
418.      C      ICO=IUC(IC)
419.      C      JF=-1
420.      C      DO 1 JC=1,JJC
421.      C      JF=JF+2
422.      C      Q(ICO+JC)=Q(ICO+JC)-Q(IFO+JF)
423.      C      1 CONTINUE
424.      C      RETURN
425.      C      END
426.      C
427.      C

```

===== APX-B-PFASMD =====

```

1.      C      *****
2.      C
3.      C      THIS PROGRAM SOLVES THE PROBLEM OF POROUS FLOW THROUGH A
4.      C      RECTANGULAR DAM OF HEIGHT Y1 AND WIDTH A.
5.      C      THE RESERVOIR TO THE RIGHT OF THE DAM IS OF HEIGHT Y2.
6.      C
7.      C      WRITTEN BY ACHI BRANDT AND COLIN CRYER AUGUST 1980
8.      C
9.      C      THIS PROGRAM WAS USED TO COMPUTE THE RESULTS IN
10.     C      SECTION 5 AND TABLE 6.4 OF THE MRC REPORT.
11.     C
12.     C      ADDITIONAL PARAMETERS USED ARE:
13.     C      NX0      THE NUMBER OF GRID INTERVALS IN THE X-DIRECTION IN
14.     C                  THE COARSEST GRID, GRID 1.
15.     C      NY0      THE NUMBER OF GRID INTERVALS IN THE Y-DIRECTION IN
16.     C                  THE COARSEST GRID, GRID 1.
17.     C      H0       THE GRID SIZE IN THE COARSEST GRID, GRID 1.
18.     C      M        THE NUMBER OF GRIDS TO BE USED.
19.     C      TOL      THE TOLERANCE. COMPUTATION TERMINATES IF THE RESIDUAL
20.     C                  ON THE FINEST GRID IS LESS THAN TOL.
21.     C      WMAX     THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
22.     C                  FINEST GRID. COMPUTATION TERMINATES WHEN WMAX IS EXCEEDED.
23.     C                  IN PRACTICAL CASES, ONE SETS WMAX=30. IN THE PRESENT WORK,
24.     C                  WE OFTEN SET WMAX=100 SO AS TO OBSERVE THE ASYMPTOTIC
25.     C                  BEHAVIOR OF THE ALGORITHM.
26.     C      MPRINT   THE GRID TO BE PRINTED AT THE END OF THE COMPUTATION.
27.     C                  THAT IS, WE PRINT THE MPRINT SUBSET OF THE FINAL ANSWER
28.     C                  ON THE GRID M.
29.     C      NQSIZE    SIZE OF ARRAY Q
30.     C                  MUST BE CHANGED FOR LARGE PROBLEMS BY EDITING PROGRAM
31.     C                  =18000 FOR DAM PROBLEM M=2,3,4,5,6
32.     C                  =70000 FOR DAM PROBLEM M=7
33.     C
34.     C      SWITCHES
35.     C
36.     C      NFGSW      =1 DAM PROBLEM
37.     C                  =2 PROBLEM (5.3),(5.4).
38.     C
39.     C
40.     C      NINTSW     =1 INJECTION. SUBROUTINE INTADD
41.     C                  =2 MODIFICATION #6. SUBROUTINE INTADM
42.     C                      CORRECTION ONLY ADDED WHEN U.NE.0. SEE (5.15).
43.     C                  =3 MODIFICATION #1. SUBROUTINE INTAPR
44.     C                      PHI=MAX(0,U)
45.     C
46.     C      NPUTSW      =1 INJECTION. SUBROUTINES PUTU AND SUBTRC
47.     C                  =2 MODIFICATION #2. SUBROUTINES PUTUNN AND SUBTNN.
48.     C                      TRANSFER 0 IF ANY NEIGHBOR ZERO.
49.     C
50.     C      NRELW       =1 NORMAL RELAXATION. SUBROUTINE RELAX
51.     C                  =2 MODIFICATION #3. SUBROUTINE RELXFR
52.     C                      VALUES OF U CHANGED ON GRID
53.     C                      K<M ONLY IF U>0 ON GRID M.
54.     C
55.     C      NRESSW      =1 INJECTION. SUBROUTINE RESCAL
56.     C                  =2 MODIFICATION #5. SUBROUTINE RESCL1
57.     C                      USES WEIGHTED RESIDUALS NEAR BOUNDARY.

```

===== APX-B-PFASMD =====

```

58.      C      RESIDUALS WITH U<0 SET EQUAL TO ZERO
59.      C      =3 MODIFICATION #4. SUBROUTINE RESCAV
60.      C      USES WEIGHTED RESIDUALS .
61.      C
62.      C      ALL THE PARAMETERS ARE SET IN THE PROGRAM, BUT THEIR VALUES
63.      C      CAN BE RESET ON THE NAMELIST INPUT CARD WHICH IS READ IN
64.      C      BY THE PROGRAM.
65.      C      THE NAMELIST CARD MUST BE PROVIDED AS INPUT.
66.      C
67.      C      THE PROGRAM SETS UP STORAGE FOR THE SOLUTIONS AND RIGHT
68.      C      HAND SIDES.
69.      C      THE SOLUTIONS ARE STORED IN ARRAYS 1 TO M.
70.      C      THE RIGHT HAND SIDES ( OR, SOMETIMES THE RESIDUALS )
71.      C      ARE STORED IN ARRAYS M+1 TO 2*M.
72.      C
73.      C      *****
74.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
75.      C      EXTERNAL G,F
76.      C      COMMON /PRBDAT/Y1,Y2,A,R
77.      C      COMMON /QDAT/NQSIZE,NQERR
78.      C      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
79.      C      NAMELIST /INDAT/Y1,Y2,A,R,NX0,NY0,H0,M,TOL,WMAX,MPRINT
80.      C      *,NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
81.      C      CHARACTER ITITLE(80)
82.      C
83.      C      READ IN AND PRINT TITLE CARDS
84.      C      FINISH READING TITLE WHEN LAST CARD IS BLANK
85.      C      FINISH RUN WHEN TITLE CARD IS BLANK
86.      C      NC=0
87.      5      READ 10,(ITITLE(I),I=1,80)
88.      10     FORMAT(80A1)
89.      C      NC=NC+1
90.      C      PRINT 11,(ITITLE(I),I=1,80)
91.      11     FORMAT(1H ,80A1)
92.      C      DO 12 I=1,80
93.      C      IF (ITITLE(I).NE.' ')GOTO 5
94.      12     CONTINUE
95.      C      IF(NC.EQ.1) STOP
96.      C
97.      C      NQSIZE=18000
98.      C      NFGSW=1
99.      C      NINTSW=1
100.     C      NPUTSW=1
101.     C      NRELSW=1
102.     C      NRESSW=1
103.     C      Y1=24
104.     C      Y2=4
105.     C      A=16
106.     C      R=32.D0/15.D0
107.     C      NX0=4
108.     C      NY0=6
109.     C      H0=4.
110.     C      M=3
111.     C      TOL=2.D-8
112.     C      WMAX=30.
113.     C      MPRINT=1
114.     C      READ(5,INDAT)

```

===== APX-B-PFASMD =====

```

115.      WRITE(6,INDAT)
116.      C      PRINT MODIFICATION NUMBERS
117.      PRINT 100
118.      100      FORMAT( '0 *** THE FOLLOWING MODIFICATIONS WERE USED *** '/')
119.      IF(NINTSW.EQ.2) PRINT 106
120.      IF(NINTSW.EQ.3) PRINT 101
121.      IF(NPUTSW.EQ.2) PRINT 102
122.      IF(NRELSW.EQ.2) PRINT 103
123.      IF(NRESSW.EQ.2) PRINT 105
124.      IF(NRESSW.EQ.3) PRINT 104
125.      101      FORMAT('0', 'MODIFICATION NUMBER 1')
126.      102      FORMAT('0', 'MODIFICATION NUMBER 2')
127.      103      FORMAT('0', 'MODIFICATION NUMBER 3')
128.      104      FORMAT('0', 'MODIFICATION NUMBER 4')
129.      105      FORMAT('0', 'MODIFICATION NUMBER 5')
130.      106      FORMAT('0', 'MODIFICATION NUMBER 6')
131.      PRINT 110
132.      110      FORMAT( ' ***** ')
133.      C      SET TIME TO ZERO
134.      CALL URTIMS(0.0)
135.      CALL PFASMD(NX0,NY0,H0,M,TOL,WMAX,G,F)
136.      C      PRINT ELAPSED TIME
137.      T=URTIMG('ELAPSED TIME')
138.      CALL SOLPRT(M,MPRINT)
139.      STOP
140.      END
141.      C
142.      C
143.      DOUBLE PRECISION FUNCTION F(X,Y)
144.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
145.      COMMON /PRBDAT/Y1,Y2,A,R
146.      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
147.      C      THIS SUBROUTINE COMPUTES THE RIGHT HAND SIDE OF THE
148.      C      GOVERNING POISSON EQUATION DEL*DEL U=F.
149.      GOTO( 1,2),NFGSW
150.      C
151.      C      DAM PROBLEM
152.      1      CONTINUE
153.      F=1.
154.      RETURN
155.      C
156.      C      PROBLEM OF SECTION 5: (5.3) AND (5.4)
157.      2      CONTINUE
158.      D=2.5*R
159.      A=DMAX1(0.D0,D-R*X-Y)
160.      B=X+Y
161.      C=2*(R**2+1)
162.      F=(C-2.*A*A)*DCOS(B) +4*(R+1)*A*DSIN(B)+2*C
163.      RETURN
164.      END
165.      C
166.      C
167.      DOUBLE PRECISION FUNCTION G(X,Y)
168.      C      THIS SUBROUTINE COMPUTES THE BOUNDARY DATA AND THE
169.      C      INITIAL APPROXIMATION TO THE SOLUTION U.
170.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
171.      COMMON /PRBDAT/Y1,Y2,A,R

```

===== APX-B-PFASMD =====

```

172.      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
173.      GOTO( 1,2),NFGSW
174.      C
175.      C      DAM PROBLEM
176.      C      THE INITIAL APPROXIMATION IS OBTAINED BY LINEAR INTERPOLATION
177.      C      IN THE X-DIRECTION BETWEEN THE GIVEN BOUNDARY DATA.
178.      1  CONTINUE
179.      G1=.5*(Y1-Y)**2
180.      G2=.5*(Y2-Y)**2
181.      IF( Y.GE.Y2) G2=0
182.      G=(G1*(A-X)+ G2*X)/A
183.      RETURN
184.      C
185.      C      PROBLEM OF SECTION 5: (5.3) AND (5.4)
186.      C      INITIAL APPROXIMATION IS A PERTURBATION OF EXACT SOLUTION
187.      2  CONTINUE
188.      D=2.5*R
189.      A=D*MAX1(0.D0,D-R*X-Y)
190.      B=X+Y
191.      G=A*A*(DCOS(B)+2)
192.      G=G+X*(3-X)*Y*(2-Y)*10
193.      RETURN
194.      END
195.      C
196.      C
197.      SUBROUTINE PFASMD(NX0,NY0,H0,M,TOL,WMAX,U1,F)
198.      C      THIS SUBROUTINE IS THE MAIN MULTIGRID SUBROUTINE.
199.      C      IT INITIALIZES THE PROBLEM, AND REPEATEDLY CALLS
200.      C      THE SUBROUTINES RELAX,RESCAL,PUTU,CORSRE,SUBTRC,AND INTADD.
201.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
202.      COMMON /QDAT/NQSIZE,NQERR
203.      EXTERNAL U1,F
204.      DIMENSION EPS(10)
205.      C
206.      C
207.      C      SET UP ARRAYS 1 TO M FOR THE SOLUTIONS
208.      C      AND ARRAYS M+1 TO 2*M FOR THE RIGHT HAND SIDES,
209.      C      AND CHECK THAT Q ARRAY IS LARGE ENOUGH
210.      NQERR=0
211.      DO 1 K=1,M
212.      K2=2**(K-1)
213.      CALL GRDFN(K,NX0*K2+1,NY0*K2+1,H0/K2)
214.      1 CALL GRDFN(K+M,NX0*K2+1,NY0*K2+1,H0/K2)
215.      PRINT 10,NQSIZE
216.      10  FORMAT(' SIZE OF Q ARRAY = ', I10)
217.      IF(NQERR.EQ.0)GOTO 12
218.      PRINT 11,NQERR
219.      11  FORMAT(' *** ERROR IN GRDFN *** ARRAY Q NOT LARGE ENOUGH ***',
220.      * /,' ARRAY Q SIZE SHOULD BE AT LEAST =', I10)
221.      STOP
222.      12  CONTINUE
223.      C
224.      C
225.      C      INITIALIZE
226.      EPS(M)=TOL
227.      K=M
228.      WU=0

```

===== APX-B-PFASMD =====

```

229.      CALL PUTF(M,U1,0)
230.      CALL PUTF(2*M,F,2)
231.      ETA=.5
232.      DELTA=.15
233.      C
234.      C      START OF MAIN LOOP IN WHICH ONE GAUSS-SEIDEL PROJECTED
235.      C      SWEEP ON GRID K IS MADE.
236.      C
237.      5 ERR=1.E30
238.      3 ERRP=ERR
239.      CALL RELSW(K,K+M,ERR)
240.      IF(WU .LE. 0) ERRBEG=ERR
241.      WU=WU+4.** (K-M)
242.      WRITE(6,4)K,ERR,WU
243.      4 FORMAT(' LEVEL',I2,' RESIDUAL NORM=', D10.3,' WORK=', F7.3)
244.      IF(ERR.LT.EPS(K))GOTO 2
245.      IF (WU.GE.WMAX)RETURN
246.      IF(K.EQ.1.OR.ERR/ERRP.LT. ETA)GO TO 3
247.      C
248.      C      GO TO COARSER GRID
249.      IF( K.NE.M .OR. WU.LE.3 ) GOTO 92
250.      FMU=0.0
251.      IF( ERR.GT.0 ) FMU=(ERR/ERRBEG)**(1.D0/(WU-1))
252.      PRINT 91,FMU
253.      91  FORMAT(' ', 20('*'),'END OF CYCLE',20('*'),'MU = ',F8.4)
254.      92  CONTINUE
255.      CALL RESSW(K,K+M,K+M-1)
256.      EPS(K-1)=DELTA*ERR
257.      K=K-1
258.      CALL PUTSW(K+1,K)
259.      CALL CORSRE(K,K+M)
260.      GOTO 5
261.      C
262.      C      GO TO FINER GRID
263.      2 IF (K.EQ.M)RETURN
264.      CALL SUBSW(K+1,K)
265.      CALL INTSW(K,K+1)
266.      K=K+1
267.      GOTO 5
268.      END
269.      C
270.      C
271.      SUBROUTINE CORSRE(K,KRHS)
272.      C      APPLIES THE DIFFERENCE OPERATOR ON GRID K
273.      C      TO THE GRID FUNCTION IN ARRAY K, AND ADDS THE RESULT TO THE
274.      C      VALUES IN ARRAY KRHS.
275.      C      KRHS      KRHS      K  K,0
276.      C      B      = R      + A  U
277.      C
278.      C      THE RESULT IS STORED IN ARRAY KRHS.
279.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
280.      COMMON Q(18000),IST(200),IRHS(200)
281.      CALL KEY(K,IST,II,JJ,H)
282.      CALL KEY(KRHS,IRHS,II,JJ,H)
283.      I1=II-1
284.      J1=JJ-1
285.      DO 1 I=2,I1

```

===== APX-B-PFASMD =====

```

286.      IR=IRHS(I)
287.      IO=IST(I)
288.      IM=IST(I-1)
289.      IP=IST(I+1)
290.      DO 1 J=2,J1
291.      A=-Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
292.      1 Q(IR+J)=-A-4.*Q(IO+J)
293.      RETURN
294.      END
295.      C
296.      C
297.      SUBROUTINE GRDFN(N,IMAX,JMAX,HH)
298.      C      SETS UP ARRAY N.
299.      C      IMAX      THE DIMENSION IN THE X DIRECTION
300.      C      JMAX      THE DIMENSION IN THE Y DIRECTION
301.      C      HH        THE GRID SIZE
302.      C      THE ARRAY NST CONTAINS THE STARTING ADDRESSES OF THE ARRAYS.
303.      C      THE ARRAY IMX CONTAINS THE MAXIMUM ROW NUMBERS
304.      C      THE ARRAY JMX CONTAINS THE MAXIMUM COL NUMBERS
305.      C      THE ARRAY H   CONTAINS THE GRID SIZES.
306.      C
307.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
308.      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
309.      COMMON /QDAT/NQSIZE,NQERR
310.      DATA IQ/1/
311.      NST(N)=IQ
312.      IMX(N)=IMAX
313.      JMX(N)=JMAX
314.      H(N)=HH
315.      IQ=IQ+IMAX*JMAX
316.      IF(IQ.LE.NQSIZE+1) RETURN
317.      NQERR=IQ-1
318.      END
319.      C
320.      C
321.      SUBROUTINE INTSW(KC,KF)
322.      C      INTERPOLATES CORRECTION ON COARSE GRID KC
323.      C      AND ADDS TO SOLUTION ON GRID KF.
324.      C      KF      KF      KC      KF      KF
325.      C      U      = PHI( I      W      + U      , U      )
326.      C      KC
327.      C
328.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
329.      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
330.      GOTO(1,2,3),NINTSW
331.      C
332.      1 CALL INTADD(KC,KF)
333.      RETURN
334.      C
335.      2 CALL INTADM(KC,KF)
336.      RETURN
337.      C
338.      3 CALL INTAPR(KC,KF)
339.      RETURN
340.      END
341.      C
342.      C

```

----- APX-B-PFASMD -----

```

343.      SUBROUTINE INTADD(KC,KF)
344.      C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
345.      C      AND ADDS TO SOLUTION ON GRID KF.
346.      C      KF      KF KC      KF      KF
347.      C      U      = PHI( I      W      + U      ; U      )
348.      C      KC
349.      C
350.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
351.      COMMON Q(18000),ISTC(200),ISTF(200)
352.      CALL KEY(KC,ISTC,IIC,JJC,HC)
353.      CALL KEY(KF,ISTF,IIF,JJF,HF)
354.      DO 1 IC=2,IIC
355.      IF=2*IC-1
356.      JF=1
357.      IFO=ISTF(IF)
358.      IFM=ISTF(IF-1)
359.      ICO=ISTC(IC)
360.      ICM=ISTC(IC-1)
361.      DO 1 JC=2,JJC
362.      JF=JF+2
363.      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
364.      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
365.      Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
366.      Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
367.      Q(IFO+JF-1)=Q(IFO+JF-1)+A
368.      1 Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
369.      RETURN
370.      END
371.      C
372.      SUBROUTINE INTADM(KC,KF)
373.      C      MODIFICATION #6.
374.      C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
375.      C      AND ADDS TO SOLUTION ON GRID KF.
376.      C      CORRECTION ONLY ADDED IF SOLUTION U ON FINE GRID IS
377.      C      NOT ZERO. SEE (5.15).
378.      C      KF      KF KC      KF
379.      C      U      = I      U      + U
380.      C      KC
381.      C
382.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
383.      COMMON Q(18000),ISTC(200),ISTF(200)
384.      CALL KEY(KC,ISTC,IIC,JJC,HC)
385.      CALL KEY(KF,ISTF,IIF,JJF,HF)
386.      DO 1 IC=2,IIC
387.      IF=2*IC-1
388.      JF=1
389.      IFO=ISTF(IF)
390.      IFM=ISTF(IF-1)
391.      ICO=ISTC(IC)
392.      ICM=ISTC(IC-1)
393.      DO 1 JC=2,JJC
394.      JF=JF+2
395.      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
396.      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
397.      IF(Q(IFO+JF).NE.0)Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
398.      IF(Q(IFM+JF).NE.0)Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
399.      IF(Q(IFO+JF-1).NE.0)Q(IFO+JF-1)=Q(IFO+JF-1)+A

```

----- APX-B-PFASMD -----

```

400.      IF(Q(IFM+JF-1).NE.0)Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
401.      1      CONTINUE
402.      RETURN
403.      END
404.      C
405.      C
406.      C
407.      SUBROUTINE INTAPR(KC,KF)
408.      C      MODIFICATION #1, PHI=MAX(0,U)
409.      C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
410.      C      AND ADDS TO SOLUTION ON GRID KF.
411.      C      KF      KF      KC      KF      KF
412.      C      U      = PHI( I      W      + U      ; U      )
413.      C      KC
414.      C
415.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
416.      C      COMMON Q(18000),ISTC(200),ISTF(200)
417.      C      CALL KEY(KC,ISTC,IIC,JJC,HC)
418.      C      CALL KEY(KF,ISTF,IIF,JJF,HF)
419.      C      DO 1 IC=2,IIC
420.      C      IF=2*IC-1
421.      C      JF=1
422.      C      IFO=ISTF(IF)
423.      C      IFM=ISTF(IF-1)
424.      C      ICO=ISTC(IC)
425.      C      ICM=ISTC(IC-1)
426.      C      DO 1 JC=2,JJC
427.      C      JF=JF+2
428.      C      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
429.      C      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
430.      C      Q(IFO+JF) =AMAX1(0.0D0, Q(IFO+JF)+Q(ICO+JC) )
431.      C      Q(IFM+JF) =AMAX1(0.0D0, Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC)) )
432.      C      Q(IFO+JF-1)=AMAX1(0.0D0,Q(IFO+JF-1)+A )
433.      C      1 Q(IFM+JF-1) =AMAX1(0.0D0, Q(IFM+JF-1)+.5*(A+AM) )
434.      C      RETURN
435.      C      END
436.      C
437.      C      SUBROUTINE KEY(K,IST,IMAX,JMAX,HH)
438.      C      RECOVERS THE INFORMATION ABOUT ARRAY K SET UP BY
439.      C      THE SUBROUTINE GRDFN.
440.      C      THE VALUE OF THE GRID FUNCTION AT THE POINT (I,J)
441.      C      IS ADDRESSED AS U(IST(J)+I).
442.      C
443.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
444.      C      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
445.      C      DIMENSION IST(1)
446.      C      IMAX=IMX(K)
447.      C      JMAX=JMX(K)
448.      C      IS=NST(K)-JMAX-1
449.      C      DO 1 I=1,IMAX
450.      C      IS=IS + JMAX
451.      C      1 IST(I)=IS
452.      C      HH=H(K)
453.      C      RETURN
454.      C      END
455.      C
456.      C

```

===== APX-B-PFASMD =====

```

457.      SUBROUTINE PUTF(K,F,NH)
458.      C      INSERTS THE VALUES OF THE FUNCTION F
459.      C      EVALUATED AT THE POINTS OF GRID K
460.      C      AND MULTIPLIED BY GRIDSIZE**NH
461.      C      INTO THE ARRAY K.
462.      C
463.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
464.      COMMON Q(18000),IST(600)
465.      CALL KEY (K,IST,II,JJ,H)
466.      H2=H**NH
467.      DO 1 I=1,II
468.      DO 1 J=1,JJ
469.      X=(I-1)*H
470.      Y=(J-1)*H
471.      1 Q(IST(I)+J)=F(X,Y)*H2
472.      RETURN
473.      END
474.      C
475.      C
476.      SUBROUTINE PUTSW(KF,KC)
477.      C      THIS SUBROUTINE TRANSFERS THE SOLUTION ON THE FINE GRID
478.      C      KF INTO THE COARSE GRID KC.
479.      C      KC,0      KC  KF
480.      C      U      = I      U
481.      C      KC
482.      C
483.      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
484.      GOTO(1,2),NPUTSW
485.      1 CALL PUTU(KF,KC)
486.      RETURN
487.      2 CALL PUTUNN(KF,KC)
488.      RETURN
489.      END
490.      C
491.      C
492.      SUBROUTINE PUTU(KF,KC)
493.      C      THIS SUBROUTINE INJECTS THE SOLUTION ON THE FINE GRID
494.      C      KF INTO THE COARSE GRID KC.
495.      C      KC,0      KC  KF
496.      C      U      = I      U
497.      C      KC
498.      C
499.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
500.      COMMON Q(18000),IUF(200),IUC(200)
501.      CALL KEY(KF,IUF,IIF,JJF,HF)
502.      CALL KEY(KC,IUC,IIC,JJC,HC)
503.      DO 1 IC=1,IIC
504.      IF=2*IC-1
505.      IFO=IUF(IF)
506.      ICO=IUC(IC)
507.      JF=-1
508.      DO 1 JC=1,JJC
509.      JF=JF+2
510.      Q(ICO+JC)=      Q(IFO+JF)
511.      1 CONTINUE
512.      RETURN
513.      END

```

===== APX-B-PFASMD =====

```

514.      C
515.      C
516.      SUBROUTINE PUTUNN(KF,KC)
517.      C      MODIFICATION #2. TRANSFER 0 IF ANY NEIGHBOR ZERO.
518.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
519.      COMMON Q(18000),IUF(200),IUC(200)
520.      CALL KEY(KF,IUF,IIF,JJF,HF)
521.      CALL KEY(KC,IUC,IIC,JJC,HC)
522.      DO 1 IC=1,IIC
523.      IF=2*IC-1
524.      IFO=IUF(IF)
525.      ICO=IUC(IC)
526.      JF=-1
527.      DO 1 JC=1,JJC
528.      JF=JF+2
529.      QTEMP=          Q(IFO+JF)
530.      IF (IC.EQ.1 .OR. IC.EQ.IIC) GO TO 1
531.      IF (JC.EQ.1 .OR. JC.EQ.JJC) GO TO 1
532.      IFP=IUF(IF+1)
533.      IFM=IUF(IF-1)
534.      IF(Q(IFP+JF-1).LE.0) QTEMP=0
535.      IF(Q(IFP+JF+1).LE.0) QTEMP=0
536.      IF(Q(IFP+JF).LE.0) QTEMP=0
537.      IF(Q(IFM+JF-1).LE.0) QTEMP=0
538.      IF(Q(IFM+JF+1).LE.0) QTEMP=0
539.      IF(Q(IFM+JF).LE.0) QTEMP=0
540.      IF(Q(IFO+JF-1).LE.0) QTEMP=0
541.      IF(Q(IFO+JF+1).LE.0) QTEMP=0
542.      1      Q(ICO+JC)=QTEMP
543.      RETURN
544.      END
545.      C
546.      C
547.      C
548.      SUBROUTINE RELSW(K,KRHS,ERR)
549.      C      CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
550.      C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
551.      C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
552.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
553.      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
554.      GOTO (1,2),NRELSW
555.      C
556.      1 CALL RELAX(K,KRHS,ERR)
557.      RETURN
558.      C
559.      2 CALL RELXFR(K,KRHS,ERR)
560.      RETURN
561.      END
562.      C
563.      C
564.      SUBROUTINE RELAX(K,KRHS,ERR)
565.      C      NORMAL RELAXATION
566.      C      CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
567.      C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
568.      C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
569.      C
570.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

```

===== APX-B-PFASMD =====

```

571.      COMMON Q(18000),IST(200),IRHS(200)
572.      CALL KEY(K,IST,II,JJ,H)
573.      CALL KEY(KRHS,IRHS,II,JJ,H)
574.      I1=II-1
575.      J1=JJ-1
576.      ERR=0.
577.      DO 1 I=2,I1
578.      IR=IRHS(I)
579.      IO=IST(I)
580.      IM=IST(I-1)
581.      IP=IST(I+1)
582.      DO 1 J=2,J1
583.      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
584.      QT=-.25*A
585.      QN=MAX(0.0,QT)
586.      ERR=ERR+(QN-Q(IO+J))**2
587.      1 Q(IO+J)=QN
588.      ERR=SQRT(ERR)/H
589.      RETURN
590.      END

591.      C
592.      SUBROUTINE RELXFR(K,KRHS,ERR)
593.      C      "FROZEN" RELAXATION: MODIFICATION # 3
594.      C      CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
595.      C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
596.      C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
597.      C      DOES NOT CHANGE VALUE OF U ON GRID K
598.      C      IF K<M AND U=0 ON GRID M
599.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
600.      COMMON Q(18000),IST(200),IRHS(200)
601.      DIMENSION ISTM(100)
602.      C      ASSUMES THAT U AND RHS ARE STORED ON GRIDS SEPARATED BY M
603.      M=KRHS-K
604.      CALL KEY(K,IST,II,JJ,H)
605.      CALL KEY(M,ISTM,IIM,JJM,HM)
606.      INTERV=2**(M-K)
607.      CALL KEY(KRHS,IRHS,II,JJ,H)
608.      I1=II-1
609.      J1=JJ-1
610.      ERR=0.
611.      DO 1 I=2,I1
612.      IR=IRHS(I)
613.      IO=IST(I)
614.      IZM=ISTM(1+INTERV*(I-1))
615.      IM=IST(I-1)
616.      IP=IST(I+1)
617.      DO 1 J=2,J1
618.      IF(K.EQ.M) GO TO 10
619.      QM=Q(IZM+1+INTERV*(J-1))
620.      IF(QM.EQ.0) GO TO 1
621.      10 CONTINUE
622.      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
623.      QT=-.25*A
624.      QN=MAX(0.0,QT)
625.      ERR=ERR+(QN-Q(IO+J))**2
626.      Q(IO+J)=QN
627.      1 CONTINUE

```

----- APX-B-PFASMD -----

```

628.      ERR=SQRT(ERR)/H
629.      RETURN
630.      END
631.      C
632.      SUBROUTINE RESSW(KF,KRF,KRC)
633.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
634.      C      IN ARRAY KRF , AND TRANSFERS INTO ARRAY KRC.
635.      C      BEFORE TRANSFER, THE RESIDUAL IS SCALED
636.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
637.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
638.      C      GRIDSIZE ON GRID KC.
639.      C      KRC      KC      KRF      KF      KF
640.      C      R      = 4*S      ( B      - A      U      )
641.      C      KF
642.      C
643.      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
644.      GOTO (1,2,3),NRESSW
645.      C
646.      1 CALL RESCAL(KF,KRF,KRC)
647.      RETURN
648.      C
649.      2 CALL RESCL1(KF,KRF,KRC)
650.      RETURN
651.      C
652.      3 CALL RESCAV(KF,KRF,KRC)
653.      RETURN
654.      END
655.      C
656.      C
657.      SUBROUTINE RESCAL(KF,KRF,KRC)
658.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
659.      C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
660.      C      BEFORE INJECTION, THE RESIDUAL IS SCALED
661.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
662.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
663.      C      GRIDSIZE ON GRID KC.
664.      C      KRC      KC      KRF      KF      KF
665.      C      R      = 4*S      ( B      - A      U      )
666.      C      KF
667.      C
668.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
669.      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
670.      CALL KEY(KF,IUF,IIF,JJF,HF)
671.      CALL KEY(KRF,IRF,IIF,JJF,HF)
672.      CALL KEY(KRC,IRC,IIC,JJC,HC)
673.      IIC1=IIC-1
674.      JJC1=JJC-1
675.      DO 1 IC=2,IIC1
676.      ICR=IRC(IC)
677.      IF=2*IC-1
678.      JF=1
679.      IFR=IRF(IF)
680.      IFO=IUF(IF)
681.      IFM=IUF(IF-1)
682.      IFP=IUF(IF+1)
683.      DO 1 JC=2,JJC1
684.      JF=JF+2

```

===== APX-B-PFASMD =====

```

685.      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
686.      1 Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
687.      RETURN
688.      END
689.      C
690.      C
691.      SUBROUTINE RESCL1(KF,KRF,KRC)
692.      C      MODIFICATION #5   UPDATED JUNE 23 1980
693.      C      USES WEIGHTED RESIDUALS NEAR THE BOUNDARY
694.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
695.      C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
696.      C      BEFORE INJECTION, THE RESIDUAL IS SCALED
697.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
698.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
699.      C      GRIDSIZE ON GRID KC.
700.      C      KRC      KC      KRF      KF      KF
701.      C      R      = 4*I      ( B      - A      U      )
702.      C      KF
703.      C
704.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
705.      C      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
706.      C      DIMENSION R(9)
707.      C      CALL KEY(KF,IUF,IIF,JJF,HF)
708.      C      CALL KEY(KRF,IRF,IIF,JJF,HF)
709.      C      CALL KEY(KRC,IRC,IIC,JJC,HC)
710.      C      IIC1=IIC-1
711.      C      JJC1=JJC-1
712.      C      DO 1 IC=2,IIC1
713.      C      ICR=IRC(IC)
714.      C      IF=2*IC-1
715.      C      JF=1
716.      C      IFR=IRF(IF)
717.      C      IFO=IUF(IF)
718.      C      IFM=IUF(IF-1)
719.      C      IFP=IUF(IF+1)
720.      C      DO 1 JC=2,JJC1
721.      C      JF=JF+2
722.      C      IF(Q(IFO+JF).EQ.0)GOTO 2
723.      C      IF(Q(IFP+JF+1).GT.0 .AND. Q(IFP+JF-1).GT.0 .AND.
724.      C      *      Q(IFO+JF+1).GT.0 .AND. Q(IFO+JF-1).GT.0 .AND.
725.      C      *      Q(IFM+JF+1).GT.0 .AND. Q(IFM+JF-1).GT.0 .AND.
726.      C      *      Q(IFM+JF ).GT.0 .AND. Q(IFP+JF ).GT.0 )GOTO 2
727.      C      N=0
728.      C      DO 3 I1=1,3
729.      C      I=IF+I1-2
730.      C      DO 3 J1=1,3
731.      C      J=JF+J1-2
732.      C      N=N+1
733.      C      IR=IRF(I)
734.      C      IO=IUF(I)
735.      C      IM=IUF(I-1)
736.      C      IP=IUF(I+1)
737.      C      S=Q(IO+J+1)+Q(IO+J-1)+Q(IM+J)+Q(IP+J)
738.      C      S=Q(IR+J)+4*Q(IO+J)-S
739.      C      IF(Q(IO+J).EQ.0)S=0
740.      C      R(N)=S
741.      C      3      CONTINUE

```

===== APX-B-PFASMD =====

```

742.      Q(ICR+JC)=R(5)+.5*(R(2)+R(4)+R(6)+R(8)+
743.      1      .5*(R(1)+R(3)+R(7)+R(9) ))
744.      GOTO 1
745.      2      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
746.      Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
747.      1      CONTINUE
748.      RETURN
749.      END
750.      C
751.      C
752.      SUBROUTINE RESCAV(KF,KRF,KRC)
753.      C      MODIFICATION #4
754.      C      AVERAGES RESIDUALS OVER NEIGHBOURING POINTS
755.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
756.      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
757.      CALL KEY(KF,IUF,IIF,JJF,HF)
758.      CALL KEY(KRF,IRF,IIF,JJF,HF)
759.      CALL KEY(KRC,IRC,IIC,JJC,HC)
760.      C      CLEAR COARSE GRID
761.      DO 9 I=1,IIC
762.      IRC=IRC(I)
763.      DO 9 J=1,JJC
764.      9      Q(ICR+J)=0.
765.      C
766.      IIF1=IIF-1
767.      JJF1=JJF-1
768.      DO 100 IF=2,IIF1
769.      IC=(IF+1)/2
770.      IL=IF+1-2*IC
771.      ICR=IRC(IC)
772.      IFR=IRF(IF)
773.      IFO=IUF(IF)
774.      IFM=IUF(IF-1)
775.      IFP=IUF(IF+1)
776.      DO 100 JF=2,JJF1
777.      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
778.      RES=(Q(IFR+JF)-S+4.*Q(IFO+JF))
779.      JC=(JF+1)/2
780.      JL=JF+1-2*JC
781.      K=2*IL+JL+1
782.      GO TO (1,2,3,4),K
783.      1      Q(ICR+JC)=Q(ICR+JC)+RES
784.      GO TO 100
785.      2      RES=RES/2
786.      Q(ICR+JC)=Q(ICR+JC)+RES
787.      Q(ICR+JC+1)=Q(ICR+JC+1)+RES
788.      GO TO 100
789.      3      RES=RES/2
790.      Q(ICR+JC)=Q(ICR+JC)+RES
791.      ICR1=IRC(IC+1)
792.      Q(ICR1+JC)=Q(ICR1+JC)+RES
793.      GO TO 100
794.      4      RES=RES/4
795.      Q(ICR+JC)=Q(ICR+JC)+RES
796.      Q(ICR+JC+1)=Q(ICR+JC+1)+RES
797.      ICR1=IRC(IC+1)
798.      Q(ICR1+JC)=Q(ICR1+JC)+RES

```

===== APX-B-PFASMD =====

```

799.      Q(ICR1+JC+1)=Q(ICR1+JC+1)+RES
800.      GO TO 100
801.      100  CONTINUE
802.      RETURN
803.      END
804.      C
805.      C
806.      SUBROUTINE SOLPRT(M,MPRINT)
807.      C      PRINTS THE ARRAY M ON THE SUBARRAY MPRINT.
808.      C
809.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
810.      COMMON Q(18000),IST(600)
811.      DIMENSION QTEM(100)
812.      CALL KEY (M,IST,II,JJ,H)
813.      INTERV=2**(M-MPRINT)
814.      DO 20 J=JJ,1,-INTERV
815.      L=0
816.      DO 10 I=1,II,INTERV
817.      C      X AND Y ARE NOT PRINTED HERE, BUT ARE COMPUTED IN
818.      C      CASE A LATER VERSION NEEDS THEM.
819.      X=(I-1)*H
820.      Y=(J-1)*H
821.      L=L+1
822.      QTEM(L)=Q(IST(I)+J)
823.      10  CONTINUE
824.      PRINT *,(QTEM(LL),LL=1,L)
825.      20  CONTINUE
826.      RETURN
827.      END
828.      C
829.      C
830.      SUBROUTINE SUBSW(KF,KC)
831.      C      THIS SUBROUTINE COMPUTES THE VALUE TRANSFERRED FROM GRID KF TO
832.      C      GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
833.      C      KC      KC      KC      KF
834.      C      W      = U      - I      U
835.      C                      KF
836.      C
837.      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
838.      GOTO(1,2),NPUTSW
839.      1  CALL SUBTRC(KF,KC)
840.      RETURN
841.      2  CALL SUBTNN(KF,KC)
842.      RETURN
843.      END
844.      C
845.      C
846.      SUBROUTINE SUBTRC(KF,KC)
847.      C      THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
848.      C      GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
849.      C      KC      KC      KC      KF
850.      C      W      = U      - I      U
851.      C                      KF
852.      C
853.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
854.      COMMON Q(18000),IUF(200),IUC(200)
855.      CALL KEY(KF,IUF,IIF,JJF,HF)

```

===== APX-B-PFASMD =====

```

856.      CALL KEY(KC,IUC,IIC,JJC,HC)
857.      DO 1 IC=1,IIC
858.      IF=2*IC-1
859.      IFO=IUF(IF)
860.      ICO=IUC(IC)
861.      JF=-1
862.      DO 1 JC=1,JJC
863.      JF=JF+2
864.      Q(ICO+JC)=Q(ICO+JC)-Q(IFO+JF)
865.      1 CONTINUE
866.      RETURN
867.      END

868.      C
869.      C
870.      SUBROUTINE SUBTNN(KF,KC)
871.      C      MODIFICATION #2. TRANSFER 0 IF ANY NEIGHBOR ZERO.
872.      C      THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
873.      C      GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
874.      C      KC      KC      KC      KF
875.      C      W      = U      - I      U
876.      C                      KF
877.      C
878.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
879.      COMMON Q(18000),IUF(200),IUC(200)
880.      CALL KEY(KF,IUF,IIF,JJF,HF)
881.      CALL KEY(KC,IUC,IIC,JJC,HC)
882.      DO 1 IC=1,IIC
883.      IF=2*IC-1
884.      IFO=IUF(IF)
885.      ICO=IUC(IC)
886.      JF=-1
887.      DO 1 JC=1,JJC
888.      JF=JF+2
889.      QTEMP=          Q(IFO+JF)
890.      IF (IC.EQ.1 .OR. IC.EQ.IIC) GO TO 1
891.      IF (JC.EQ.1 .OR. JC.EQ.JJC) GO TO 1
892.      IFP=IUF(IF+1)
893.      IFM=IUF(IF-1)
894.      IF(Q(IFP+JF-1).LE.0) QTEMP=0
895.      IF(Q(IFP+JF+1).LE.0) QTEMP=0
896.      IF(Q(IFP+JF).LE.0) QTEMP=0
897.      IF(Q(IFM+JF-1).LE.0) QTEMP=0
898.      IF(Q(IFM+JF+1).LE.0) QTEMP=0
899.      IF(Q(IFM+JF).LE.0) QTEMP=0
900.      IF(Q(IFO+JF-1).LE.0) QTEMP=0
901.      IF(Q(IFO+JF+1).LE.0) QTEMP=0
902.      1 Q(ICO+JC)=Q(ICO+JC)-QTEMP
903.      RETURN
904.      END

905.      C
906.      C

```

===== APPX-C-PFMG =====

```

1.      C      *****
2.      C
3.      C      THIS PROGRAM SOLVES THE PROBLEM OF POROUS FLOW THROUGH A
4.      C      RECTANGULAR DAM OF HEIGHT Y1 AND WIDTH A.
5.      C      THE RESERVOIR TO THE RIGHT OF THE DAM IS OF HEIGHT Y2.
6.      C
7.      C      WRITTEN BY ACHI BRANDT AND COLIN CRYER AUGUST 1980
8.      C
9.      C      THIS PROGRAM WAS USED TO COMPUTE THE RESULTS IN
10.     C      SECTION 6 OF THE MRC REPORT.
11.     C
12.     C      ADDITIONAL PARAMETERS USED ARE:
13.     C      NX0      THE NUMBER OF GRID INTERVALS IN THE X-DIRECTION IN
14.     C                  THE COARSEST GRID, GRID 1.
15.     C      NY0      THE NUMBER OF GRID INTERVALS IN THE Y-DIRECTION IN
16.     C                  THE COARSEST GRID, GRID 1.
17.     C      H0       THE GRID SIZE IN THE COARSEST GRID, GRID 1.
18.     C      M        THE NUMBER OF GRIDS TO BE USED.
19.     C      LIN      THE STARTING GRID. LIN.GE.2
20.     C      TOL      THE TOLERANCE
21.     C      RATIO    TOLERANCE ON GRID L IS TOLL=TOL*RATIO**L
22.     C      WMAXM    THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
23.     C                  FINEST GRID. COMPUTATION TERMINATES WHEN WMAXM IS
24.     C                  EXCEEDED.
25.     C      WMAX      THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
26.     C                  GRID L<M. COMPUTATION ON GRID L TERMINATES WHEN WMAX IS
27.     C                  EXCEEDED.
28.     C      MPRINT   THE GRID TO BE PRINTED AT THE END OF THE COMPUTATION.
29.     C                  THAT IS, WE PRINT THE MPRINT SUBSET OF THE FINAL ANSWER
30.     C                  ON THE GRID M.
31.     C      NQSIZE    SIZE OF ARRAY Q
32.     C                  MUST BE CHANGED FOR LARGE PROBLEMS BY EDITING PROGRAM
33.     C                  =18000 FOR DAM PROBLEM M=2,3,4,5,6
34.     C      NR1      AFTER NR1 RELAXATIONS ON THE GRID K+1 THERE IS A
35.     C                  TRANSFER TO GRID K.
36.     C      NR2      AFTER A TOTAL NUMBER OF NR2 RELAXATIONS ON GRID K
37.     C                  THERE IS A TRANSFER TO GRID K+1
38.     C      NCYC      MAXIMUM NUMBER OF CYCLES ON LEVEL L, LIN< L<M
39.     C      NCYCLN    MAXIMUM NUMBER OF CYCLES ON LEVEL LIN
40.     C      NCYCM     MAXIMUM NUMBER OF CYCLES ON LEVEL M
41.     C      ETA      IF ERR.GE.ETA*ERRP GO TO COARSER GRID
42.     C      DELTA     EPS(K-1)=DELTA*(ERROR ERR ON GRID K)
43.     C      PREC      EPS(L)=MAX(PREC*TAU(L-1),TOL*RATIO**L)
44.     C      PRECM     EPS(M)=MAX(PRECM*TAU(M-1),TOL*RATIO**M)
45.     C
46.     C      WE CAN ALSO DO TAU EXTRAPOLATION:
47.     C      ITAU      IF ITAU=1 DO TAU EXTRAPOLATION
48.     C      PT        ORDER OF EXTRAPOLATION
49.     C
50.     C      SWITCHES
51.     C
52.     C      NFGSW      USED IN SUBROUTINES F,G,SOLRED
53.     C      NFGSW      =1 DAM PROBLEM
54.     C                  =2 PROBLEM (5.3),(5.4).
55.     C
56.     C
57.     C      NINTSW     =1 INJECTION. SUBROUTINE INTADD

```

===== APPX-C-PFMG =====

```

58.      C          =2 MODIFICATION #6. SUBROUTINE INTADM
59.      C          CORRECTION ONLY ADDED WHEN U.NE.0. SEE (5.15).
60.      C
61.      C      NRESSW =1 INJECTION. SUBROUTINE RESCAL
62.      C          =2 MODIFICATION #5. SUBROUTINE RESCL1
63.      C          USES WEIGHTED RESIDUALS NEAR BOUNDARY.
64.      C          RESIDUALS WITH U<0 SET EQUAL TO ZERO
65.      C
66.      C
67.      C      ALL THE PARAMETERS ARE SET IN THE PROGRAM, BUT THEIR VALUES
68.      C      CAN BE RESET ON THE NAMELIST INPUT CARD WHICH IS READ IN
69.      C      BY THE PROGRAM.
70.      C      THE NAMELIST CARD MUST BE PROVIDED AS INPUT.
71.      C
72.      C      THE PROGRAM SETS UP STORAGE FOR THE SOLUTIONS AND RIGHT
73.      C      HAND SIDES.
74.      C      THE SOLUTIONS ARE STORED IN ARRAYS 1 TO M.
75.      C      THE RIGHT HAND SIDES ( OR, SOMETIMES THE RESIDUALS )
76.      C      ARE STORED IN ARRAYS M+1 TO 2*M.
77.      C
78.      C      THE EXACT SOLUTION (WHEN KNOWN) IS STORED IN GRID NGRSOL
79.      C      THE VALUES OF TAU ARE STORED IN GRIDS 2M+1 TO 3M-1
80.      C
81.      C      *****
82.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
83.      C      EXTERNAL G,F
84.      C      COMMON /PRBDAT/Y1,Y2,A,R
85.      C      COMMON /QDAT/NQSIZE,NQERR
86.      C      COMMON /SOLTAU/M,NGRSOL,PT
87.      C      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
88.      C      NAMELIST /INDAT/NX0,NY0,H0,M,LIN,NR1,NR2,ETA,DELTA
89.      C      * ,TOL,RATIO,PREC,PRECM,WMAX,WMAXM,NCYC,NCYCLN,NCYCM,ITAU,PT,
90.      C      * MPRINT,Y1,Y2,A,R
91.      C      * ,NFGSW,NINTSW,NRELSW,NRESSW
92.      C      CHARACTER ITITLE(80)
93.      C
94.      C      READ IN AND PRINT TITLE CARDS
95.      C      FINISH READING TITLE WHEN LAST CARD IS BLANK
96.      C      FINISH RUN WHEN TITLE CARD IS BLANK
97.      C      PRINT 18
98.      C      18 FORMAT(1H1)
99.      C      NC=0
100.     C      5 READ 10,(ITITLE(I),I=1,80)
101.     C      10 FORMAT(80A1)
102.     C      NC=NC+1
103.     C      PRINT 11,(ITITLE(I),I=1,80)
104.     C      11 FORMAT(1H ,80A1)
105.     C      DO 12 I=1,80
106.     C      IF (ITITLE(I).NE.' ')GOTO 5
107.     C      12 CONTINUE
108.     C      IF(NC.EQ.1) STOP
109.     C
110.     C      NQSIZE=18000
111.     C      NFGSW=1
112.     C      NINTSW=1
113.     C      NRESSW=1
114.     C      Y1=24

```

===== APPX-C-PFMG =====

```

115.      Y2=4
116.      A=16
117.      R=32.D0/15.D0
118.      NX0=2
119.      NY0=3
120.      H0=8.
121.      M=6
122.      LIN=2
123.      NR1=2
124.      NR2=3
125.      ETA=10.
126.      DELTA=0
127.      TOL=0
128.      RATIO=1
129.      PREC=0
130.      PRECM=1
131.      WMAX=30.
132.      WMAXM=40
133.      NCYC=1
134.      NCYCLN=3
135.      NCYCM=10
136.      ITAU=0
137.      PT =2
138.      MPRINT=2
139.      READ(5,INDAT)
140.      WRITE(6,INDAT)
141.      C      PRINT MODIFICATION NUMBERS
142.      PRINT 100
143.      100    FORMAT( '0 *** THE FOLLOWING MODIFICATIONS WERE USED *** '/')
144.      IF(NINTSW.EQ.2) PRINT 106
145.      IF(NINTSW.EQ.3) PRINT 101
146.      IF(NRELSW.EQ.2) PRINT 103
147.      IF(NRESSW.EQ.2) PRINT 105
148.      IF(NRESSW.EQ.3) PRINT 104
149.      101    FORMAT('0', 'MODIFICATION NUMBER 1')
150.      103    FORMAT('0', 'MODIFICATION NUMBER 3')
151.      104    FORMAT('0', 'MODIFICATION NUMBER 4')
152.      105    FORMAT('0', 'MODIFICATION NUMBER 5')
153.      106    FORMAT('0', 'MODIFICATION NUMBER 6')
154.      PRINT 110
155.      110    FORMAT( ' ***** ')
156.      C      SET TIME TO ZERO
157.      CALL URTIMS(0.0)
158.      CALL PFMG(NX0,NY0,H0,LIN,NR1,NR2,ETA,DELTA
159.      *      ,TOL,RATIO,PREC,PRECM,WMAX,WMAXM,NCYC,NCYCLN,NCYCM,ITAU,
160.      *      MPRINT,G,F)
161.      T=URTIMG('ELAPSE')
162.      19      FORMAT(1H0, ' GRID-M SOLUTION',//)
163.      PRINT 19
164.      CALL SOLPRT(M,MPRINT)
165.      PRINT 20
166.      20      FORMAT(1H1, ' GRID-7 SOLUTION',//)
167.      CALL SOLPRT(NGRSOL,MPRINT)
168.      STOP
169.      END
170.      C
171.      C

```

===== APPX-C-PFMG =====

```

172.      DOUBLE PRECISION FUNCTION F(X,Y)
173.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
174.      COMMON /PRBDAT/Y1,Y2,A,R
175.      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
176.      C      THIS SUBROUTINE COMPUTES THE RIGHT HAND SIDE OF THE
177.      C      GOVERNING POISSON EQUATION DEL*DEL U=F.
178.      GOTO( 1,2),NFGSW
179.      C
180.      C      DAM PROBLEM
181.      1  CONTINUE
182.      F=1.
183.      RETURN
184.      C
185.      C      PROBLEM OF SECTION 5: (5.3) AND (5.4)
186.      2  CONTINUE
187.      D=2.5*R
188.      A=D*MAX1(0.D0,D-R*X-Y)
189.      B=X+Y
190.      C=2*(R**2+1)
191.      F=(C-2.*A*A)*DCOS(B) +4*(R+1)*A*DSIN(B)+2*C
192.      RETURN
193.      END
194.      C
195.      C
196.      DOUBLE PRECISION FUNCTION G(X,Y)
197.      C      THIS SUBROUTINE COMPUTES THE BOUNDARY DATA AND THE
198.      C      INITIAL APPROXIMATION TO THE SOLUTION U.
199.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
200.      COMMON /PRBDAT/Y1,Y2,A,R
201.      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
202.      GOTO( 1,2),NFGSW
203.      C
204.      C      DAM PROBLEM
205.      C      THE INITIAL APPROXIMATION IS OBTAINED BY LINEAR INTERPOLATION
206.      C      IN THE X-DIRECTION BETWEEN THE GIVEN BOUNDARY DATA.
207.      1  CONTINUE
208.      G1=.5*(Y1-Y)**2
209.      G2=.5*(Y2-Y)**2
210.      IF( Y.GE.Y2) G2=0
211.      G=(G1*(A-X)+ G2*X)/A
212.      RETURN
213.      C
214.      C      PROBLEM OF SECTION 5: (5.3) AND (5.4)
215.      C      INITIAL APPROXIMATION IS A PERTURBATION OF EXACT SOLUTION
216.      2  CONTINUE
217.      D=2.5*R
218.      A=D*MAX1(0.D0,D-R*X-Y)
219.      B=X+Y
220.      G=A*A*(DCOS(B)+2)
221.      G=G+X*(3-X)*Y*(2-Y)*10
222.      RETURN
223.      END
224.      C
225.      SUBROUTINE PFMG(NX0,NY0,H0,LIN,NR1,NR2,ETA,DELTA
226.      *      ,TOL,RATIO,PREC,PRECM,WMAX,WMAXM,NCYC,NCYCLN,NCYCM,ITAU,
227.      *      MPRINT,U1,F)
228.      C      THIS SUBROUTINE IS THE MAIN FULL MULTIGRID SUBROUTINE.

```

===== APPX-C-PFMG =====

```

229.      C      IT INITIALIZES THE PROBLEM, AND REPEATEDLY CALLS
230.      C      THE SUBROUTINES RELAX, RESCAL, PUTU, CORSRE, SUBTRC, AND INTADD.
231.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
232.      C      COMMON /QDAT/NQSIZE,NQERR
233.      C      EXTERNAL U1,F
234.      C      DIMENSION EPS(10),IR2(10)
235.      C      COMMON /SOLTAU/M,NGRSOL,PT
236.      C
237.      C
238.      C      SET UP ARRAYS 1 TO M FOR THE SOLUTIONS
239.      C      AND ARRAYS M+1 TO 2*M FOR THE RIGHT HAND SIDES,
240.      C      AND ARRAYS 2M+1 TO 3M-1 FOR TAU ARRAYS AND
241.      C      SET ASIDE SPACE FOR GRID-7 SOLUTION IN 3M=NGRSOL GRID
242.      C      AND CHECK THAT Q ARRAY IS LARGE ENOUGH
243.      C      NQERR=0
244.      C      DO 1 K=1,M
245.      C      K2=2**(K-1)
246.      C      CALL GRDFN(K,NX0*K2+1,NY0*K2+1,H0/K2)
247.      C      CALL GRDFN(K+M,NX0*K2+1,NY0*K2+1,H0/K2)
248.      C      1 CALL GRDFN(K+2*M,NX0*K2+1,NY0*K2+1,H0/K2)
249.      C      NGRSOL=3*M
250.      C      PRINT 90,NQSIZE
251.      C      90      FORMAT(' SIZE OF Q ARRAY = ', I10)
252.      C      IF(NQERR.EQ.0)GOTO 92
253.      C      PRINT 91,NQERR
254.      C      91      FORMAT(' *** ERROR IN GRDFN *** ARRAY Q NOT LARGE ENOUGH ***',
255.      C      * /, ' ARRAY Q SIZE SHOULD BE AT LEAST =', I10)
256.      C      STOP
257.      C      92 CONTINUE
258.      C
259.      C
260.      C      CALL SOLRED
261.      C
262.      C      INITIALIZE
263.      C      WU=0
264.      C      CALL PUTF(LIN,U1,0)
265.      C      DO 10 L=LIN,M
266.      C
267.      C      BEGIN NEW FINEST LEVEL
268.      C
269.      C      PRINT 6,L
270.      C      6      FORMAT(1H0,60(1H.),I3,2X,60(1H.)/)
271.      C      CALL PUTF(L+M,F,2)
272.      C      TOLL=TOL*(RATIO**L)
273.      C      EPS(L)=TOLL
274.      C      WU=.25*WU
275.      C      NCYCL=NCYC
276.      C      IF(L.EQ.M)NCYCL=NCYCM
277.      C      ICYC=0
278.      C      WMAXL=WMAX
279.      C      IF(L.EQ.M)WMAXL=WMAXM
280.      C      PRECL=PREC
281.      C      IF(L.EQ.M)PRECL=PRECM
282.      C
283.      C
284.      C      K=L
285.      C      IR2(L)=0

```

===== APPX-C-PFMG =====

```

286.      C
287.      C      BEGIN A NEW WORK LEVEL
288.      C
289.      5  IR1=0
290.      ERR=1.E30
291.      C
292.      C      RELAX ONCE ON GRID K
293.      3  ERRP=ERR
294.      CALL RELAX(K,K+M,ERR)
295.      WU=WU+4.*(K-L)
296.      IR1=IR1+1
297.      IR2(K)=IR2(K)+1
298.      WRITE(6,40)K,ERR,WU,IR1,IR2(K)
299.      40  FORMAT(' LEVEL',I2,' RESIDUAL NORM=', D10.3,' WORK=', F7.3
300.      *    , ' IR1= ',I2,' IR2(K)=',I2)
301.      C
302.      C      DECIDE WHICH GRID TO USE NEXT
303.      IF (WU.GE.WMAXL)GOTO 20
304.      IF(ERR.LT.EPS(K))GOTO 2
305.      IF(IR2(K).NE.NR2)GOTO 8
306.      IF( K.LT.L)GOTO 2
307.      C
308.      ICYC=ICYC+1
309.      IF(ICYC.EQ.NCYCL .AND. L.NE.LIN)GOTO 20
310.      IF(ICYC.EQ.NCYCLN .AND. L.EQ.LIN)GOTO 20
311.      IR2(L)=0
312.      IR1=0
313.      C
314.      8  IF(IR1.EQ.NR1)GOTO 4
315.      IF(IR1.EQ.1.OR.ERR.LT. ERRP*ETA)GO TO 3
316.      C
317.      C      GO TO COARSER GRID
318.      4  IF(K.EQ.1)GOTO 3
319.      CALL RESSW(K,K+M,K+M-1)
320.      CALL RESBW(K,K+M,K+2*M-1)
321.      EPS(K-1)=DELTA*ERR
322.      K=K-1
323.      CALL PUTU(K+1,K)
324.      CALL CORSRE(K,K+M)
325.      ITAUEX=0
326.      IF((ITAU.EQ.1).AND.(L.GT.LIN).AND.(K.EQ.L-1))ITAUEX=1
327.      CALL TAUCAM(K,K+M,K+2*M,ITAUEX,TAUGNM)
328.      PRINT 60,TAUGNM,K
329.      60  FORMAT(50X,'GREEN NORM OF TAU-Z =',E12.3,5X,'K=',I2)
330.      IF(K.EQ.(L-1))EPS(L)=DMAX1(PRECL*TAUGNM,TOLL)
331.      IR2(K)=0
332.      GOTO 5
333.      C
334.      C      GO TO FINER GRID
335.      2  IF(K.EQ.L)GOTO 20
336.      CALL SUBTRC(K+1,K)
337.      CALL INTSW(K,K+1)
338.      K=K+1
339.      GOTO 5
340.      C
341.      C
342.      C      FINISHED WITH LEVEL L

```

===== APPX-C-PFMG =====

```

343.      20      CONTINUE
344.      C
345.      C      THE NEXT SEVEN STATEMENTS COMPUTE THE GREEN NORM OF TAU
346.      C      AND THE GREEN AND L-INFINITY NORMS OF THE ERROR
347.      C      ( IF ACCURATE SOLUTION IS KNOWN )
348.      11      CALL RESSW(L,L+M,L+M-1)
349.      CALL RESBW(L,L+M,L+2*M-1)
350.      CALL PUTU(L,L-1)
351.      CALL CORSRE(L-1,L-1+M)
352.      CALL TAUCAM(L-1,L-1+M,L-1+2*M,0,TAUGNM)
353.      K=L-1
354.      PRINT 60,TAUGNM,K
355.      CALL DIFFMX(L)
356.      C
357.      C
358.      IF(L.EQ.M)GOTO 10
359.      CALL INTRP3(L,L+1)
360.      CALL PUTB(U1,L+1)
361.      C
362.      10      CONTINUE
363.      RETURN
364.      END
365.      C
366.      C
367.      SUBROUTINE CORSRE(K,KRHS)
368.      C      APPLIES THE DIFFERENCE OPERATOR ON GRID K
369.      C      TO THE GRID FUNCTION IN ARRAY K, AND ADDS THE RESULT TO THE
370.      C      VALUES IN ARRAY KRHS.
371.      C      KRHS      KRHS      K K,0
372.      C      B      = R      + A U
373.      C
374.      C      THE RESULT IS STORED IN ARRAY KRHS.
375.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
376.      C      COMMON Q(18000),IST(200),IRHS(200)
377.      C      CALL KEY(K,IST,II,JJ,H)
378.      C      CALL KEY(KRHS,IRHS,II,JJ,H)
379.      C      I1=II-1
380.      C      J1=JJ-1
381.      C      DO 1 I=2,I1
382.      C      IR=IRHS(I)
383.      C      IO=IST(I)
384.      C      IM=IST(I-1)
385.      C      IP=IST(I+1)
386.      C      DO 1 J=2,J1
387.      C      A=-Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
388.      C      1 Q(IR+J)=-A-4.*Q(IO+J)
389.      C      RETURN
390.      C      END
391.      C
392.      C
393.      SUBROUTINE DIFFMX(K)
394.      C      NOT TIMED
395.      C      COMPARES SOLUTION ON GRID K WITH ACCURATE SOLUTION
396.      C      STORED IN GRID NGRSOL
397.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
398.      C      COMMON Q(18000),IST(200),ISTA(200)
399.      C      COMMON /SOLTAU/M,NGRSOL,PT

```

===== APPX-C-PFMG =====

```

400.      TIME=URTIMG(0)
401.      CALL KEY(K,IST,II,JJ,H)
402.      CALL KEY(NGRSOL,ISTA,IIA,JJA,HA)
403.      DIFMX=0.
404.      DIFGNM=0
405.      SOLMX=0.
406.      SOLGNM=0
407.      INTERV=(IIA-1)/(II-1)
408.      DO 1 I=1,II
409.      X=(I-1)*H
410.      IA=(I-1)*INTERV+1
411.      DO 1 J=1,JJ
412.      Y=(J-1)*H
413.      JA=(J-1)*INTERV+1
414.      DIF=ABS(Q(ISTA(IA)+JA) -Q(IST(I)+J))
415.      DIFGNM=DIFGNM+DIF*DIF
416.      SOL=ABS(Q(ISTA(IA)+JA))
417.      SOLGNM=SOLGNM+SOL*SOL
418.      SOLMX=AMAX1(SOL,SOLMX)
419.      1 DIFMX=AMAX1(DIF,DIFMX)
420.      DIFGNM=SQRT(DIFGNM)/H
421.      PRINT 101,DIFMX,DIFGNM
422.      101 FORMAT(15X,' SOLUTION ERROR: L INFINITY NORM =',E13.5,
423.      * 5X,'GNORM = ',E13.5)
424.      SOLGNM=SQRT(SOLGNM)/H
425.      PRINT 102,SOLMX,SOLGNM
426.      102 FORMAT(15X,' SOLUTION          : L INFINITY NORM =',E13.5,
427.      * 5X,'GNORM = ',E13.5)
428.      PRINT 103,DIFMX/SOLMX,DIFGNM/SOLGNM
429.      103 FORMAT(15X,' RELATIVE ERROR: L INFINITY NORM =',E13.5,
430.      * 5X,'GNORM = ',E13.5)
431.      CALL URTIMS(TIME)
432.      RETURN
433.      END
434.      C
435.      C
436.      SUBROUTINE GRDFN(N,IMAX,JMAX,HH)
437.      C      SETS UP ARRAY N.
438.      C      IMAX      THE DIMENSION IN THE X DIRECTION
439.      C      JMAX      THE DIMENSION IN THE Y DIRECTION
440.      C      HH        THE GRID SIZE
441.      C      THE ARRAY NST CONTAINS THE STARTING ADDRESSES OF THE ARRAYS.
442.      C      THE ARRAY IMX CONTAINS THE MAXIMUM ROW NUMBERS
443.      C      THE ARRAY JMX CONTAINS THE MAXIMUM COL NUMBERS
444.      C      THE ARRAY H   CONTAINS THE GRID SIZES.
445.      C
446.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
447.      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
448.      COMMON /QDAT/NQSIZE,NQERR
449.      DATA IQ/1/
450.      NST(N)=IQ
451.      IMX(N)=IMAX
452.      JMX(N)=JMAX
453.      H(N)=HH
454.      IQ=IQ+IMAX*JMAX
455.      IF(IQ.LE.NQSIZE+1) RETURN
456.      NQERR=IQ-1

```

APPX-C-PFMG =====

```

457.      END
458.      C
459.      C
460.      SUBROUTINE INTSW(KC,KF)
461.      C      INTERPOLATES CORRECTION ON COARSE GRID KC
462.      C      AND ADDS TO SOLUTION ON GRID KF.
463.      C      KF      KF KC      KF      KF
464.      C      U      = PHI( I      W      + U      ; U      )
465.      C      KC
466.      C
467.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
468.      C      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
469.      C      GOTO(1,2),NINTSW
470.      C
471.      1 CALL INTADD(KC,KF)
472.      C      RETURN
473.      C
474.      2 CALL INTADM(KC,KF)
475.      C      RETURN
476.      C      END
477.      C
478.      C
479.      SUBROUTINE INTADD(KC,KF)
480.      C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
481.      C      AND ADDS TO SOLUTION ON GRID KF.
482.      C      KF      KF KC      KF      KF
483.      C      U      = PHI( I      W      + U      ; U      )
484.      C      KC
485.      C
486.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
487.      C      COMMON Q(18000),ISTC(200),ISTF(200)
488.      C      CALL KEY(KC,ISTC,IIC,JJC,HC)
489.      C      CALL KEY(KF,ISTF,IIF,JJF,HF)
490.      C      DO 1 IC=2,IIC
491.      C      IF=2*IC-1
492.      C      JF=1
493.      C      IFO=ISTF(IF)
494.      C      IFM=ISTF(IF-1)
495.      C      ICO=ISTC(IC)
496.      C      ICM=ISTC(IC-1)
497.      C      DO 1 JC=2,JJC
498.      C      JF=JF+2
499.      C      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
500.      C      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
501.      C      Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
502.      C      Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
503.      C      Q(IFO+JF-1)=Q(IFO+JF-1)+A
504.      C      1 Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
505.      C      RETURN
506.      C      END
507.      C
508.      SUBROUTINE INTADM(KC,KF)
509.      C      MODIFICATION #6.
510.      C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
511.      C      AND ADDS TO SOLUTION ON GRID KF.
512.      C      CORRECTION ONLY ADDED IF SOLUTION U ON FINE GRID IS
513.      C      NOT ZERO. SEE (5.15).

```

=====APPX-C-PFMG =====

```

514.      C      KF      KF      KC      KF
515.      C      U      = I      U      + U
516.      C              KC
517.      C
518.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
519.      COMMON Q(18000),ISTC(200),ISTF(200)
520.      CALL KEY(KC,ISTC,IIC,JJC,HC)
521.      CALL KEY(KF,ISTF,IIF,JJF,HF)
522.      DO 1 IC=2,IIC
523.      IF=2*IC-1
524.      JF=1
525.      IFO=ISTF(IF)
526.      IFM=ISTF(IF-1)
527.      ICO=ISTC(IC)
528.      ICM=ISTC(IC-1)
529.      DO 1 JC=2,JJC
530.      JF=JF+2
531.      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
532.      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
533.      IF(Q(IFO+JF).NE.0)Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
534.      IF(Q(IFM+JF).NE.0)Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
535.      IF(Q(IFO+JF-1).NE.0)Q(IFO+JF-1)=Q(IFO+JF-1)+A
536.      IF(Q(IFM+JF-1).NE.0)Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
537.      1 CONTINUE
538.      RETURN
539.      END
540.      C
541.      C
542.      C
543.      SUBROUTINE INTRP3(KC,KF)
544.      C      PERFORMS CUBIC INTERPOLATION
545.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
546.      COMMON Q(18000),IUF(200),IUC(200)
547.      CALL KEY(KF,IUF,IIF,JJF,HF)
548.      CALL KEY(KC,IUC,IIC,JJC,HC)
549.      C
550.      C      KF      KF      KC      KF
551.      C      U      = J      U      + U
552.      C              KC
553.      C
554.      C      INTERPOLATE IN COARSE COLUMNS USING COARSE COLUMN DATA
555.      DO 20 IC=1,IIC
556.      IF=2*IC-1
557.      IFO=IUF(IF)
558.      ICO=IUC(IC)
559.      Q(IFO+1)=Q(ICO+1)
560.      C      FIRST POINT IN COLUMN. USE EQU (6.3)
561.      Q(IFO+2)=(5*Q(ICO+1)+15*Q(ICO+2)-5*Q(ICO+3)+Q(ICO+4))/16
562.      JJC2=JJC-2
563.      DO 10 JC=2,JJC2
564.      JF=2*JC-1
565.      Q(IFO+JF)=Q(ICO+JC)
566.      C
567.      C      INTERIOR POINT IN COLUMN. USE EQU (6.2)
568.      Q(IFO+JF+1)=(-Q(ICO+JC-1)+9*Q(ICO+JC)
569.      *      +9*Q(ICO+JC+1)-Q(ICO+JC+2))/16.
570.      10 CONTINUE

```

===== APPX-C-PFMG =====

```

571.      Q(IFO+JJF-2)=Q(ICO+JJC-1)
572.      Q(IFO+JJF-1)=(Q(ICO+JJC-3)-5*Q(ICO+JJC-2)
573.      C
574.      C      LAST POINT IN COLUMN. USE EQU (6.3)
575.      *      +15*Q(ICO+JJC-1)+5*Q(ICO+JJC))/16
576.      Q(IFO+JJF)=Q(ICO+JJC)
577.      20  CONTINUE
578.      C
579.      C      INTERPOLATE IN INTERMEDIATE FINE COLUMNS
580.      C      USING ROW DATA
581.      C
582.      C      FIRST COLUMN. USE EQU (6.3)
583.      IM1=IUF(1)
584.      IO=IUF(2)
585.      IP1=IUF(3)
586.      IP3=IUF(5)
587.      IP5=IUF(7)
588.      DO 30 J=1,JJF
589.      Q(IO+J)=(5*Q(IM1+J)+15*Q(IP1+J)
590.      *      -5*Q(IP3+J)+Q(IP5+J))/16.
591.      30  CONTINUE
592.      C
593.      C      INTERMEDIATE COLUMNS. USE EQU (6.2)
594.      IIF3=IIF-3
595.      DO 40 I=4,IIF3,2
596.      IM3=IUF(I-3)
597.      IM1=IUF(I-1)
598.      IO=IUF(I)
599.      IP1=IUF(I+1)
600.      IP3=IUF(I+3)
601.      DO 40 J=1,JJF
602.      Q(IO+J)=(-Q(IM3+J)+9*Q(IM1+J)
603.      *      +9*Q(IP1+J)-Q(IP3+J))/16.
604.      40  CONTINUE
605.      C
606.      C      LAST COLUMN. USE EQU (6.3)
607.      IM5=IUF(IIF-6)
608.      IM3=IUF(IIF-4)
609.      IM1=IUF(IIF-2)
610.      IO=IUF(IIF-1)
611.      IP1=IUF(IIF)
612.      DO 50 J=1,JJF
613.      Q(IO+J)=(Q(IM5+J)-5*Q(IM3+J)
614.      *      +15*Q(IM1+J)+5*Q(IP1+J))/16
615.      50  CONTINUE
616.      RETURN
617.      END
618.      C
619.      C
620.      SUBROUTINE KEY(K,IST,IMAX,JMAX,HH)
621.      C      RECOVERS THE INFORMATION ABOUT ARRAY K SET UP BY
622.      C      THE SUBROUTINE GRDFN.
623.      C      THE VALUE OF THE GRID FUNCTION AT THE POINT (I,J)
624.      C      IS ADDRESSED AS U(IST(J)+I).
625.      C
626.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
627.      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)

```

===== APPX-C-PFMG =====

```

628.      DIMENSION IST(1)
629.      IMAX=IMX(K)
630.      JMAX=JMX(K)
631.      IS=NST(K)-JMAX-1
632.      DO 1 I=1,IMAX
633.      IS=IS + JMAX
634.      1 IST(I)=IS
635.      HH=H(K)
636.      RETURN
637.      END
638.      C
639.      C
640.      SUBROUTINE PUTB(F,K)
641.      C      INSERTS THE BOUNDARY VALUES OF THE FUNCTION F
642.      C      EVALUATED AT THE POINTS OF GRID K
643.      C      INTO THE ARRAY K.
644.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
645.      COMMON Q(18000)
646.      DIMENSION IST(200)
647.      CALL KEY(K,IST,II,JJ,H)
648.      II1=II-1
649.      DO 1 J=1,JJ
650.      X=0.
651.      Y=(J-1)*H
652.      Q(IST(1)+J)=F(X,Y)
653.      X=(II-1)*H
654.      Q(IST(II)+J)=F(X,Y)
655.      1 CONTINUE
656.      DO 2 I=2,II1
657.      Y=0.
658.      X=(I-1)*H
659.      Q(IST(I)+1)=F(X,Y)
660.      Y=(JJ-1)*H
661.      Q(IST(I)+JJ)=F(X,Y)
662.      2 CONTINUE
663.      RETURN
664.      END
665.      C
666.      C
667.      C
668.      C
669.      SUBROUTINE PUTF(K,F,NH)
670.      C      INSERTS THE VALUES OF THE FUNCTION F
671.      C      EVALUATED AT THE POINTS OF GRID K
672.      C      AND MULTIPLIED BY GRIDSIZE**NH
673.      C      INTO THE ARRAY K.
674.      C
675.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
676.      COMMON Q(18000),IST(600)
677.      CALL KEY (K,IST,II,JJ,H)
678.      H2=H**NH
679.      DO 1 I=1,II
680.      DO 1 J=1,JJ
681.      X=(I-1)*H
682.      Y=(J-1)*H
683.      1 Q(IST(I)+J)=F(X,Y)*H2
684.      RETURN

```

===== APPX-C-PFMG =====

```

685.      END
686.      C
687.      C
688.      SUBROUTINE PUTU(KF,KC)
689.      C      THIS SUBROUTINE INJECTS THE SOLUTION ON THE FINE GRID
690.      C      KF INTO THE COARSE GRID KC.
691.      C      KC,0      KC  KF
692.      C      U      = I      U
693.      C      KF
694.      C
695.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
696.      COMMON Q(18000),IUF(200),IUC(200)
697.      CALL KEY(KF,IUF,IIF,JJF,HF)
698.      CALL KEY(KC,IUC,IIC,JJC,HC)
699.      DO 1 IC=1,IIC
700.      IF=2*IC-1
701.      IFO=IUF(IF)
702.      ICO=IUC(IC)
703.      JF=-1
704.      DO 1 JC=1,JJC
705.      JF=JF+2
706.      Q(ICO+JC)=      Q(IFO+JF)
707.      1 CONTINUE
708.      RETURN
709.      END
710.      C
711.      C
712.      SUBROUTINE RELAX(K,KRHS,ERR)
713.      C      NORMAL RELAXATION
714.      C      CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
715.      C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
716.      C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
717.      C
718.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
719.      COMMON Q(18000),IST(200),IRHS(200)
720.      CALL KEY(K,IST,II,JJ,H)
721.      CALL KEY(KRHS,IRHS,II,JJ,H)
722.      I1=II-1
723.      J1=JJ-1
724.      ERR=0.
725.      DO 1 I=2,I1
726.      IR=IRHS(I)
727.      IO=IST(I)
728.      IM=IST(I-1)
729.      IP=IST(I+1)
730.      DO 1 J=2,J1
731.      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
732.      QT=-.25*A
733.      QN=MAX(0.0,QT)
734.      ERR=ERR+(QN-Q(IO+J))**2
735.      1 Q(IO+J)=QN
736.      ERR=SQRT(ERR)/H
737.      RETURN
738.      END
739.      C
740.      SUBROUTINE RESBW(KF,KRF,KRC)
741.      C      SAME AS RESSW EXCEPT THAT ONLY THE RHS B IS TREATED

```

===== APPX-C-PFMG =====

```

742.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
743.      C      IN ARRAY KRF , AND TRANSFERS INTO ARRAY KRC.
744.      C      BEFORE TRANSFER, THE RESIDUAL IS SCALED
745.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
746.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
747.      C      GRIDSIZE ON GRID KC.
748.      C      KRC      KC      KRF
749.      C      R      = 4*S      ( B      )
750.      C      KF
751.      C
752.      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
753.      GOTO (1,2),NRESSW
754.      C
755.      1 CALL RESBAL(KF,KRF,KRC)
756.      RETURN
757.      C
758.      2 CALL RESBL1(KF,KRF,KRC)
759.      RETURN
760.      END
761.      C
762.      C
763.      SUBROUTINE RESBAL(KF,KRF,KRC)
764.      C      SAME AS RESCAL EXCEPT THAT ONLY RHS B IS TREATED
765.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
766.      C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
767.      C      BEFORE INJECTION, THE RESIDUAL IS SCALED
768.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
769.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
770.      C      GRIDSIZE ON GRID KC.
771.      C      KRC      KC      KRF
772.      C      R      = 4*S      ( B      )
773.      C      KF
774.      C
775.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
776.      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
777.      CALL KEY(KF,IUF,IIF,JJF,HF)
778.      CALL KEY(KRF,IRF,IIF,JJF,HF)
779.      CALL KEY(KRC,IRC,IIC,JJC,HC)
780.      IIC1=IIC-1
781.      JJC1=JJC-1
782.      DO 1 IC=2,IIC1
783.      ICR=IRC(IC)
784.      IF=2*IC-1
785.      JF=1
786.      IFR=IRF(IF)
787.      IFO=IUF(IF)
788.      IFM=IUF(IF-1)
789.      IFP=IUF(IF+1)
790.      DO 1 JC=2,JJC1
791.      JF=JF+2
792.      1 Q(ICR+JC)=4.*(Q(IFR+JF))
793.      RETURN
794.      END
795.      C
796.      C
797.      SUBROUTINE RESBL1(KF,KRF,KRC)
798.      C      SAME AS RESCL1 EXCEPT THAT ONLY RHS B IS TREATED

```

===== APPX-C-PFMG =====

```

799.  C      MODIFICATION #5 UPDATED JUNE 23 1980
800.  C      USES WEIGHTED RESIDUALS NEAR THE BOUNDARY
801.  C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
802.  C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
803.  C      BEFORE INJECTION, THE RESIDUAL IS SCALED
804.  C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
805.  C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
806.  C      GRIDSIZE ON GRID KC.
807.  C      KRC      KC      KRF
808.  C      R      = 4*S      ( B      )
809.  C      KF
810.  C
811.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
812.      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
813.      DIMENSION R(9)
814.      CALL KEY(KF,IUF,IIF,JJF,HF)
815.      CALL KEY(KRF,IRF,IIF,JJF,HF)
816.      CALL KEY(KRC,IRC,IIC,JJC,HC)
817.      IIC1=IIC-1
818.      JJC1=JJC-1
819.      DO 1 IC=2,IIC1
820.      ICR=IRC(IC)
821.      IF=2*IC-1
822.      JF=1
823.      IFR=IRF(IF)
824.      IFO=IUF(IF)
825.      IFM=IUF(IF-1)
826.      IFP=IUF(IF+1)
827.      DO 1 JC=2,JJC1
828.      JF=JF+2
829.      IF(Q(IFO+JF).EQ.0)GOTO 2
830.      IF(Q(IFP+JF+1).GT.0 .AND. Q(IFP+JF-1).GT.0 .AND.
831.      *   Q(IFO+JF+1).GT.0 .AND. Q(IFO+JF-1).GT.0 .AND.
832.      *   Q(IFM+JF+1).GT.0 .AND. Q(IFM+JF-1).GT.0 .AND.
833.      *   Q(IFM+JF ).GT.0 .AND. Q(IFP+JF ).GT.0 )GOTO 2
834.      N=0
835.      DO 3 I1=1,3
836.      I=IF+I1-2
837.      DO 3 J1=1,3
838.      J=JF+J1-2
839.      N=N+1
840.      IR=IRF(I)
841.      IO=IUF(I)
842.      IM=IUF(I-1)
843.      IP=IUF(I+1)
844.      S=Q(IR+J)
845.      IF(Q(IO+J).EQ.0)S=0
846.      R(N)=S
847.      3 CONTINUE
848.      Q(ICR+JC)=R(5)+.5*(R(2)+R(4)+R(6)+R(8)+
849.      *   .5*(R(1)+R(3)+R(7)+R(9) ))
850.      GOTO 1
851.      2 Q(ICR+JC)=4.*Q(IFR+JF)
852.      1 CONTINUE
853.      RETURN
854.      END
855.  C

```

===== APPX-C-PFMG =====

```

856.      C
857.      SUBROUTINE RESSW(KF,KRF,KRC)
858.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
859.      C      IN ARRAY KRF , AND TRANSFERS INTO ARRAY KRC.
860.      C      BEFORE TRANSFER, THE RESIDUAL IS SCALED
861.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
862.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
863.      C      GRIDSIZE ON GRID KC.
864.      C      KRC      KC      KRF      KF      KF
865.      C      R      = 4*S      ( B      - A      U      )
866.      C      KF
867.      C
868.      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
869.      GOTO (1,2),NRESSW
870.      C
871.      1 CALL RESCAL(KF,KRF,KRC)
872.      RETURN
873.      C
874.      2 CALL RESCL1(KF,KRF,KRC)
875.      RETURN
876.      END
877.      C
878.      C
879.      SUBROUTINE RESCAL(KF,KRF,KRC)
880.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
881.      C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
882.      C      BEFORE INJECTION, THE RESIDUAL IS SCALED
883.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
884.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
885.      C      GRIDSIZE ON GRID KC.
886.      C      KRC      KC      KRF      KF      KF
887.      C      R      = 4*S      ( B      - A      U      )
888.      C      KF
889.      C
890.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
891.      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
892.      CALL KEY(KF,IUF,IIF,JJF,HF)
893.      CALL KEY(KRF,IRF,IIF,JJF,HF)
894.      CALL KEY(KRC,IRC,IIC,JJC,HC)
895.      IIC1=IIC-1
896.      JJC1=JJC-1
897.      DO 1 IC=2,IIC1
898.      ICR=IRC(IC)
899.      IF=2*IC-1
900.      JF=1
901.      IFR=IRF(IF)
902.      IFO=IUF(IF)
903.      IFM=IUF(IF-1)
904.      IFP=IUF(IF+1)
905.      DO 1 JC=2,JJC1
906.      JF=JF+2
907.      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
908.      1 Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
909.      RETURN
910.      END
911.      C
912.      C

```

AD-A096 652

WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER

F/6 12/1

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY-ETC(U)

OCT 80 A BRANDT, C W CRYER

DAA629-80-C-0041

UNCLASSIFIED

MRC-TSR-2131

NL

2 of 2

AD-A096 652



END

DATE

FILED

4-81

DTIC

===== APPX-C-PFMG =====

```

913.      SUBROUTINE RESCL1(KF,KRF,KRC)
914.      C      MODIFICATION #5 UPDATED JUNE 23 1980
915.      C      USES WEIGHTED RESIDUALS NEAR THE BOUNDARY
916.      C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
917.      C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
918.      C      BEFORE INJECTION, THE RESIDUAL IS SCALED
919.      C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
920.      C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
921.      C      GRIDSIZE ON GRID KC.
922.      C      KRC      KC      KRF      KF      KF
923.      C      R      = 4*S      ( B      - A      U      )
924.      C      KF
925.      C
926.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
927.      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
928.      DIMENSION R(9)
929.      CALL KEY(KF,IUF,IIF,JJF,HF)
930.      CALL KEY(KRF,IRF,IIF,JJF,HF)
931.      CALL KEY(KRC,IRC,IIC,JJC,HC)
932.      IIC1=IIC-1
933.      JJC1=JJC-1
934.      DO 1 IC=2,IIC1
935.      ICR=IRC(IC)
936.      IF=2*IC-1
937.      JF=1
938.      IFR=IRF(IF)
939.      IFO=IUF(IF)
940.      IFM=IUF(IF-1)
941.      IFP=IUF(IF+1)
942.      DO 1 JC=2,JJC1
943.      JF=JF+2
944.      IF(Q(IFO+JF).EQ.0)GOTO 2
945.      IF(Q(IFP+JF+1).GT.0 .AND. Q(IFP+JF-1).GT.0 .AND.
946.      *      Q(IFO+JF+1).GT.0 .AND. Q(IFO+JF-1).GT.0 .AND.
947.      *      Q(IFM+JF+1).GT.0 .AND. Q(IFM+JF-1).GT.0 .AND.
948.      *      Q(IFM+JF ).GT.0 .AND. Q(IFP+JF ).GT.0 )GOTO 2
949.      N=0
950.      DO 3 I1=1,3
951.      I=IF+I1-2
952.      DO 3 J1=1,3
953.      J=JF+J1-2
954.      N=N+1
955.      IR=IRF(I)
956.      IO=IUF(I)
957.      IM=IUF(I-1)
958.      IP=IUF(I+1)
959.      S=Q(IO+J+1)+Q(IO+J-1)+Q(IM+J)+Q(IP+J)
960.      S=Q(IR+J)+4*Q(IO+J)-S
961.      IF(Q(IO+J).EQ.0)S=0
962.      R(N)=S
963.      3 CONTINUE
964.      Q(ICR+JC)=R(5)+.5*(R(2)+R(4)+R(6)+R(8)+
965.      1      .5*(R(1)+R(3)+R(7)+R(9) ))
966.      GOTO 1
967.      2 S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
968.      Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
969.      1 CONTINUE

```

===== APPX-C-PFMG =====

```

970.      RETURN
971.      END
972.      C
973.      C
974.      SUBROUTINE SOLPRT(K,MPRINT)
975.      C      NOT TIMED
976.      C      PRINTS THE ARRAY K ON THE SUBARRAY MPRINT.
977.      C      IF K<MPRINT, PRINTS ENTIRE ARRAY K
978.      C
979.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
980.      COMMON Q(18000),QTEM(100),IST(600)
981.      TIME=URTIMG(0)
982.      CALL KEY (MPRINT,IST,IIM,JJ,H)
983.      CALL KEY (K,IST,II,JJ,H)
984.      INTERV=1
985.      IF(K.GT.MPRINT)INTERV=(II-1)/(IIM-1)
986.      DO 20 J=JJ,1,-INTERV
987.      L=0
988.      DO 10 I=1,II,INTERV
989.      C      X AND Y ARE NOT PRINTED HERE, BUT ARE COMPUTED IN
990.      C      CASE A LATER VERSION NEEDS THEM.
991.      X=(I-1)*H
992.      Y=(J-1)*H
993.      L=L+1
994.      QTEM(L)=Q(IST(I)+J)
995.      10      CONTINUE
996.      PRINT *,(QTEM(LL),LL=1,L)
997.      20      CONTINUE
998.      CALL URTIMS(TIME)
999.      RETURN
1000.     END
1001.     C
1002.     C
1003.     C
1004.     SUBROUTINE SOLRED
1005.     C      NOT TIMED
1006.     C      PUTS ACCURATE SOLUTION INTO GRID NGRSOL
1007.     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
1008.     COMMON Q(18000),ISTA(200),QTEM(600)
1009.     COMMON /SOLTAU/M,NGRSOL,PT
1010.     COMMON /PRBDAT/Y1,Y2,A,R
1011.     COMMON /SWDAT/NFGSW,NINTSW,NRESSW
1012.     TIME=URTIMG(0)
1013.     CALL KEY(NGRSOL,ISTA,IIA,JJA,HA)
1014.     C
1015.     GOTO(1,2),NFGSW
1016.     C
1017.     C      DAM PROBLEM
1018.     C      ACCURATE SOL IS DOUBLE PRECISION ON GRID M=7
1019.     C      WITH INITIAL GRID 2X3
1020.     C      STORED IN FILE 10.
1021.     1      CONTINUE
1022.     MA=7
1023.     IIMA=2**(MA-1)*2+1
1024.     INTERV=(IIMA-1)/(IIA-1)
1025.     JJMA=(JJA-1)*INTERV+1
1026.     REWIND 10

```

=====APPX-C-PFMG=====

```

1027.      DO 20 JA=1,JJMA
1028.      READ(10) (QTEM(IA),IA=1,IIMA)
1029.      J=(JA-1)/INTERV+1
1030.      IF( (J-1)*INTERV .NE. JA-1 )GOTO 20
1031.      DO 10 I=1,IIA
1032.      IA=(I-1)*INTERV+1
1033.      Q(ISTA(I)+J)=QTEM(IA)
1034.      10 CONTINUE
1035.      20 CONTINUE
1036.      GOTO 1000
1037.      C
1038.      C
1039.      C      PROBLEM OF SECTION 5: (5.3) AND (5.4)
1040.      C      EXACT SOLUTION KNOWN
1041.      2 CONTINUE
1042.      D=2.5*R
1043.      DO 30 I=1,IIA
1044.      IO=ISTA(I)
1045.      DO 25 J=1,JJA
1046.      X=(I-1)*HA
1047.      Y=(J-1)*HA
1048.      A=DMAX1(0.D0,D-R*X-Y)
1049.      B=X+Y
1050.      G=A*A*(DCOS(B)+2)
1051.      Q(IO+J)=G
1052.      25 CONTINUE
1053.      30 CONTINUE
1054.      GOTO 1000
1055.      C
1056.      1000 CONTINUE
1057.      CALL URTIMS(TIME)
1058.      RETURN
1059.      END
1060.      C
1061.      SUBROUTINE SUBTRC(KF,KC)
1062.      C      THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
1063.      C      GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
1064.      C      KC      KC      KC      KF
1065.      C      W      = U      - I      U
1066.      C      KC
1067.      C
1068.      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
1069.      COMMON Q(18000),IUF(200),IUC(200)
1070.      CALL KEY(KF,IUF,IIF,JJF,HF)
1071.      CALL KEY(KC,IUC,IIC,JJC,HC)
1072.      DO 1 IC=1,IIC
1073.      IF=2*IC-1
1074.      IFO=IUF(IF)
1075.      ICO=IUC(IC)
1076.      JF=-1
1077.      DO 1 JC=1,JJC
1078.      JF=JF+2
1079.      Q(ICO+JC)=Q(ICO+JC)-Q(IFO+JF)
1080.      1 CONTINUE
1081.      RETURN
1082.      END
1083.      C

```

===== APPX-C-PFMG =====

```

1084.      C
1085.      SUBROUTINE TAUCAM(KU,KR,KF,ITAU,TAUGNM)
1086.      C      COMPUTES TAU AND TAU-Z GREEN NORM
1087.      C      UPDATED AUGUST 26 1980
1088.      C      PERFORMS TAU EXTRAPOLATION IF ITAU=1
1089.      C      BY ADDING TAU TO RHS ON GRID
1090.      C      GRID KU CONTAINS U
1091.      C      GRID KR CONTAINS SUM OF FIRST TWO TERMS IN (6.7)
1092.      C      PREVIOUSLY OBTAINED USING RESSW AND CORSRE
1093.      C      GRID KF CONTAINS THIRD BRACKET IN (6.7) PREVIOUSLY
1094.      C      COMPUTED BY RESBW
1095.      C      ITAU IS PARAMETER WHICH DETERMINES WHETHER EXTRAPOLATION
1096.      C      WILL BE PERFORMED
1097.      C      TAUGNM IS RETURNED AS GREEN NORM OF TAU-Z
1098.      C
1099.      C      K-1    PT      K-1    K    K K      K-1 K-1 K      K-1 K
1100.      C      T      = 2      *(4S    (B  - A U )) + (A    I    U ) - (4S    B ) )
1101.      C      ----- K                      K                      K
1102.      C      2**PT-1
1103.      C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
1104.      C      COMMON Q(18000),IKR(200),IKF(200),IKU(200)
1105.      C      COMMON /SOLTAU/M,NGRSOL,PT
1106.      C      CALL KEY(KR,IKR,II,JJ,HK)
1107.      C      CALL KEY(KF,IKF,II,JJ,HK)
1108.      C      CALL KEY(KU,IKU,II,JJ,HK)
1109.      C      A=2.**PT/(2.**PT-1)
1110.      C      TAUGNM=0
1111.      C      II1=II-1
1112.      C      JJ1=JJ-1
1113.      C      DO 1 IK=2,II1
1114.      C      IRK=IKR(IK)
1115.      C      IFKO=IKF(IK)
1116.      C      IO=IKU(IK)
1117.      C      IM=IKU(IK-1)
1118.      C      IP=IKU(IK+1)
1119.      C      DO 1 JK=2,JJ1
1120.      C      T=Q(IRK+JK)-Q(IFKO+JK)
1121.      C      T=A*T
1122.      C      IF(Q(IO+JK).EQ.0)T=0
1123.      C      TAUGNM=TAUGNM+T*T
1124.      C      IF( Q(JK+IO+1).EQ.0 .OR.
1125.      C      * Q(IO+JK-1).EQ.0 .OR. Q(IM+JK).EQ.0 .OR.
1126.      C      * Q(IP+JK).EQ.0) T=0
1127.      C      IF(ITAU.EQ.1)Q(IRK+JK)=T+Q(IFKO+JK)
1128.      C      1 CONTINUE
1129.      C      TAUGNM=SQRT(TAUGNM)/HK
1130.      C      RETURN
1131.      C      END

```

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MRZ-TSR-2131	2. GOVT ACCESSION NO. AD-A096652	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS		5. TYPE OF REPORT & PERIOD COVERED Summary report - no specific reporting period
7. AUTHOR(s) Achi Brandt ~~and~~ Colin W. Cryer		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of 610 Walnut Street Madison, Wisconsin 53706		8. CONTRACT OR GRANT NUMBER(s) 13 DAAG29-80-C-0041 NSF-MCS77-26732
11. CONTROLLING OFFICE NAME AND ADDRESS See Item 18 below.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit # 3 (Numerical Analysis and Computer Science)
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 9 T - hnizal summary rept.		12. REPORT DATE 11 October 1980
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		13. NUMBER OF PAGES 96
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
18. SUPPLEMENTARY NOTES U. S. Army Research Office P. O. Box 12211 Research Triangle Park North Carolina 27709		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Multigrid Algorithms Free Boundary Problems Linear Complementarity Problems		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) We show that the multigrid algorithms of Brandt can be adapted to solve linear complementarity problems arising from free boundary problems. The multigrid algorithms are significantly faster than previous algorithms. Using the multigrid algorithms, which are simple modifications of multigrid algorithms for equalities, it is possible to solve the difference equations to within truncation error using less work than the equivalent of six Gauss-Seidel sweeps on the finest grid.		