JOURNAL OF RESEARCH of the National Bureau of Standards
Vol. 85, No. 5, September-October 1980

The Numerical Solution of a Nonseparable Elliptic Partial Differential Equation by Preconditioned Conjugate Gradients

John Gregg Lewis* and Ronald G. Rehm†

National Bureau of Standards, Washington, D.C. 20234

April 16, 1980

In this report the combination of an iterative technique, the conjugate gradient algorithm, with a fast direct method, cyclic reduction, is used to solve the linear algebraic equations resulting from discretization of a nonseparable elliptic partial differential equation. An expository discussion of the conjugate gradient and preconditioned conjugate gradient algorithms and of their use in the solution of partial differential equations is presented. New results extending the use of the preconditioned conjugate gradients technique to singular linear equations which arise from discretized elliptic equations with Neumann boundary conditions are also given. The algorithms are applied to solve a specific elliptic equation which arises in the study of buoyant convection produced by a room fire. A code was developed to implement the algorithms for this application. Numerical results obtained through testing and use of the code are discussed.

Key Words: Conjugate gradient algorithm; elliptic partial differential equations; iterative methods for linear algebraic equations; Neumann boundary conditions; sparse matrices.

The numerical solution of the linear algebraic equations which result from a discretization of an elliptic partial differential equation has been, and continues to be, the focus of much research in numerical analysis. Over the past 25 years there have been many advances, some toward improved iterative schemes (ADI, SOR, etc.), others toward fast direct methods (most notably those based on FFT's). See Rice [1] [1] for a brief survey of the impact of these advances. In this paper we discuss the combination of an iterative technique, the conjugate gradient algorithm, with a fast direct method, cyclic reduction, to extend the capabilities of the fast solver. For examples of the use of similar combinations of algorithms, see Concus & Golub [2], Concus, Golub and O'Leary [3], O'Leary [4] and O'Leary and Widlund [5].

The work described in this paper resulted from a study of buoyant convection carried out at the National Bureau of Standards, [6, 7, 8]. The specific elliptic partial differential equation for pressure arising in this work is used as a model problem in the present paper. The experience of the first author with the buoyant convection model motivated the writing of sections 1 and 2, which gives an exposition of the conjugate gradient and preconditioned conjugate gradient algorithms. These sections contain no new material; they were written so that this paper may be accessible to an audience unfamiliar with the development of the conjugate gradient algorithm and its use in the solution of partial differential equations. Section 3 has a discussion of the model problem, the pressure equation. Section 4 contains several new results extending the use of preconditioned conjugate gradient technology to the singular linear equations which result from Neumann boundary value problems. We conclude with some numerical examples from the buoyant convection problem. A listing of a FORTRAN program implementing the algorithm discussed here is presented in [15].

*Center for Applied Mathematics, National Engineering Laboratory; and Mathematical Sciences Department, Johns Hopkins University. Current Address: Boeing Computer Services, Co., Mail Stop 9C-01, P.O. Box 24346, Seattle, Washington 98124

†Center for Applied Mathematics, National Engineering Laboratory.

[1] Figures in brackets indicate literature references at the end of this paper.

1. An introduction to the conjugate gradient algorithm

The discretization of an elliptic differential operator by standard finite difference or finite element techniques produces, as the finite analog of the continuous operator, very large matrices with most elements zero. From an algebraic standpoint we can very often view the numerical solution of the differential equation as:

$$\text{Solve } Ax = b \tag{1}$$

where A is a k by k, (k large) real symmetric, positive definite and sparse matrix.

Fast direct methods with minimal storage requirements exist when A is the finite difference operator resulting from a separable elliptic operator on a rectangular region, and for some other common, but specific, cases. If the operator is nonseparable, or the region non-rectangular, the special direct methods may not apply. Moreover, the use of general direct methods such as factoring A into a Cholesky decomposition,

$$A = LL^T,$$

often impose unbearable storage requirements because many zero entries are filled in during the decomposition.

Iterative methods, such as SOR and Gauss-Seidel, solve a transformation of the problem. Instead of solving (1) directly, they "split" A as $A = M-N$, and solve the equivalent problem through an iteration of the form $Mx^{k+1} = Nx^k + b$; i.e., find a fixed point of the equation

$$Mx = Nx + b. \tag{2}$$

The efficiency of the solution method depends in part on the appropriateness of the splitting and in part on acceleration of the iteration toward the fixed point.

The conjugate gradient algorithm can be motivated as the solution of another transformation of problem (1). Consider the inner product $<x,y> \equiv x^TAy$. This induces a norm $\|x\|_A \equiv \sqrt{x^TAx}$ on R^k when A is positive definite.
Problem (1) is equivalent to: find x to minimize

$$E(x) = (x-x^*)^T A(x-x^*) = \|x-x^*\|^2_A \tag{3}$$

where x^* is the solution of (1). The problems are equivalent because $E(x) \geqslant 0$ for all x, and $E(x) = 0 <=> x = x^*$. The immediate usefulness of the transformation is not obvious, especially since computing $E(x)$ appears to require the solution, x^*, of the problem we wish to solve. Note however, that we can evaluate whether a given x is or is not a solution by computing the residual $r = b - Ax$. Further, while we cannot evaluate $E(x)$ directly, we can evaluate its gradient vector:

$$\nabla E(x) = 2(Ax - Ax^*) = 2(Ax - b) = -2r.$$

Hence, r gives us the direction in which E decreases most rapidly (unless r is identically zero, which characterizes the solution).

These two characteristics of problem (3) lead directly to a simple algorithm for solving (3):

STEEPEST DESCENT: Given a guess x_i, not a solution,

$$\begin{cases} \text{set } r_i = b - Ax_i \\ \text{set } x_{i+1} = x_i + \alpha_i r_i \end{cases}$$

where α_i is a scalar chosen to make x_{i+1} minimize $E(x_{i+1})$ for all vectors of the form $x_i + \beta r_i$.
We can easily compute the gradient r_i. It is also easy to compute α_i so that $E(x_{i+1}) = E(x_i + \alpha_i r_i) \leqslant E(x_i + \beta r_i)$ for all β. The mimimum of $E(x_i + \beta r_i)$ is given by the solution of

$$\frac{d}{d\beta} E(x_i + \beta r_i) = 0$$

But

$$\frac{d}{d\beta} E(x_i + \beta r_i) = 2(\beta r_i^T A r_i - r_i^T r_i),$$

so

$$\alpha_i = \frac{r_i^T r_i}{r_i^T A r_i} = \frac{\|r_i\|_2^2}{\|r_i\|_A^2} \tag{4}$$

is the optimal choice for α_i. Hence, we can carry out the iteration to minimize $E(x)$ without computing E. Note also that A enters the iteration only in forming the products Ax_i and Ar_i.

Unfortunately steepest descent is not a practical algorithm because the convergence can be very slow. We can obtain more rapid convergence by using not the steepest descent directions $\{r_i\}$, but instead a sequence of downhill or descent directions $\{p_i\}$ which satisfy additional properties. We characterize the term "descent direction" and simultaneously find the optimal distance to move in that direction, through the following simple result.

LEMMA: *If* $\mathrm{p}^T\mathrm{r} \neq 0$, *then*

$$\beta = \frac{\mathrm{p}^T\mathrm{r}}{\mathrm{p}^T\mathrm{Ap}} \textit{ minimizes } \mathrm{E}(\mathrm{x} + \beta\mathrm{p}) \textit{ for all } \beta. \tag{5}$$

PROOF: Simply differentiate $E(x + \beta p)$ with respect to β:

$$\frac{d}{d\beta} E(x + \beta p) = 2(p^T A(x - x^*) + \beta p^T A p)$$
$$= 2(-p^T r + \beta p^T A p).$$

It is a simple computation to show that $E(x + \beta p) < E(x)$. Note that β is positive if $p^T r > 0$, which characterizes p as a descent, not an ascent, direction.

The first observation which leads to improved convergence is that the choice in (5) for β, not only solves the one-dimensional minimization, it gives us the $p-$ component of x^* exactly. To see this, note that A positive definite implies that we can uniquely write

$$\begin{aligned} x + \beta p &= \alpha p + \delta w \\ x^* &= \alpha^* p + \delta^* z \end{aligned} \tag{6}$$

where $w^T A p = 0 = z^T A p$. This last condition is read: w, z are A-orthogonal or *A-conjugate* to p. Now substitute the decompositions (6) into E:

$$E(x + \beta p) = (\alpha - \alpha^*)^2 p^T A p + \text{ terms not involving } \alpha, \alpha^* \text{ or } p \tag{7}$$

The choice (5) for β minimizes the left side of (7), and clearly $\alpha = \alpha^*$ minimizes the right hand side of (7). Hence choosing $\beta = \dfrac{p^T r}{p^T A p}$ implies that

$$(x + \beta p) - x^* = \delta w - \delta^* z. \tag{8}$$

The error vector is A-conjugate to p, i.e., lies in the $k-1$ dimensional subspace of vectors t such that $t^T Ap = 0$. If all succeeding descent directions are chosen from this subspace, we can preserve the exact solution of the problem in the direction p.

The second observation is that we can easily choose successive directions $p_1, p_2, p_3, \ldots$ such that the $\{p_i\}$ are all pairwise A-conjugate descent directions (hence are "A-conjugate gradients") and hence, such that the successive errors $x_1 - x^*$, $x_2 - x^*$, $x_3 - x^*$ are constrained to successively smaller dimensional subspaces. The set $\{p_i\}$, $i = 1, \ldots, k$ gives a basis for R^k and $x_k - x^*$ must be A-conjugate to all of the $\{p_i\}$, hence must be zero. Thus, the process must give the exact answer after at most k iterations. We now write out the conjugate gradient iteration directly:

CONJUGATE GRADIENTS: Given x_1, compute $r_1 = b - Ax_1$. Set the initial descent direction p_1 to be r_1. For $i = 2, 3, 4, \ldots$, do.

Move to the minimum in direction p_i and evaluate the new residual

$$\begin{aligned} \alpha_i &= \frac{r_i^T p_i}{p_i^T A p_i} \\ x_{i+1} &= x_i + \alpha_i p_i \\ r_{i+1} &= b - Ax_{i+1} \\ &= r_i - \alpha_i A p_i \end{aligned} \tag{9}$$

Compute a new descent direction A-orthogonal to all of its predecessors

$$\begin{aligned} \beta_i &= -\frac{r_{i+1}^T A p_i}{p_i^T A p_i} \\ p_{i+1} &= r_{i+1} + \beta_i p_i \end{aligned} \tag{10}$$

The initial step (9) in the iteration is the step in the steepest descent direction p_i, as discussed above. Step (10), the choice of a new descent direction, is simply a single step of Gram-Schmidt orthogonalization, to remove from the steepest descent direction its component in the direction Ap_i.

Hence

$$p_{i+1}^T A p_i = 0 \tag{11}$$

Futher the choice of α_i implies directly that

$$r_{i+1}^T p_i = 0 \tag{12}$$

The following identities are derived easily from (9) and (10):

$$\begin{aligned} r_{i+1}^T r_i &= 0; \\ r_i^T p_i &= r_i^T r_i; \\ r_i^T A p_i &= p_i^T A p_i. \end{aligned} \tag{13}$$

The most important algebraic consequences of (9) and (10) yield only to a lengthy inductive argument (not given here—see Reid [12] or Hestenes and Stiefel [13] for example)

$$\left.\begin{aligned} r_{i+1}^T r_j &= 0 \\ p_{i+1}^T A p_j &= 0 \end{aligned}\right\} \text{ for } all\; j \leqslant i \tag{14}$$

The identities in (14) imply that all of the $\{p_i\}$ are A-conjugate, even though we perform an explicit orthogonalization only to one descent direction in (10).

In the conjugate gradient iteration observe that A enters only in forming a matrix-vector product, Ap_i; there are no transformations done which could destroy the sparseness of A. The dominant cost per step is usually that of forming the product Ap_i; in this case the conjugate gradient iteration is very little more expensive per iteration than the steepest descent algorithm and it offers the guarantee of convergence within k steps.

The practical reason for using the conjugate gradient algorithm is that we often obtain satisfactory accuracy after only a very few iterations. Certain theoretical bounds for the convergence of the algorithm depend on the condition number, $\varkappa$, of A, defined by:

$$\varkappa = \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \tag{15}$$

where λ_{max} (A) is the largest eigenvalue of A and λ_{min} (A) is the smallest eigenvalue of A (Note: A is positive definite, so $\varkappa$ is positive, and at least as great as one). The following bounds can be found in the literature (see Daniel [14], for example):

$$\|x_i - x^*\|^2 \leqslant 4 \frac{E(x_i)}{\lambda_{min}} \cdot \left(\frac{\sqrt{\varkappa}-1}{\sqrt{\varkappa}+1} \right)^{2(i-1)}$$

$$E(x_i) \leqslant 4 \left(\frac{1 - \sqrt{\frac{1}{\varkappa}}}{1 + \sqrt{\frac{1}{\varkappa}}} \right)^{2(i-1)} E(x_1)$$

Clearly, convergence is rapid when $\varkappa$ is close to one (and takes place in one step if $\varkappa$ equals one). For well-conditioned problems very few iterations will suffice to obtain highly accurate solutions.

2. Preconditioned Conjugate Gradients and Matrix-Splittings.

Our basic problem is to solve

$$Ax = b, \tag{16}$$

which is the large, sparse linear system which arises from the discretization of an elliptic partial differential equation. We can assume that the matrix A is symmetric and positive definite, so the method of conjugate gradients certainly can be used. However, the condition number, $\varkappa$ (A), is usually very large for these problems; the conjugate gradient algorithm converges very slowly when applied directly to A and is not competitive with other methods.

The preconditioned conjugate gradients method arises, like SOR, from matrix splittings. Consider

$$A = M - N$$

where M is another positive-definite, symmetric matrix, but one for which we can easily solve linear systems. There exists a symmetric positive-definite matrix $M^{-1/2}$ such that

$$M^{-1} = (M^{-1/2})(M^{-1/2})$$

or

$$[(M^{-1/2})(M^{-1/2})]^{-1} = M$$

(This is the symmetric matrix whose eigenvectors are the same as M's and whose corresponding eigenvalues are the reciprocals of the square roots of the eigenvalues of M. We shall not use this form, however; we need only the formal existence of $M^{-1/2}$.) We can solve the equations $Ax = b$ by computing

$$d = M^{-1/2}b; \tag{17}$$

and solving

$$Cz = d \tag{18}$$

where $C = M^{-1/2}AM^{-1/2}$, by the conjugate gradient algorithm (Note that C is a positive-definite, symmetric matrix), finally computing

$$x = M^{-1/2}z \tag{19}$$

This transformation from $Ax = b$ to $Cy = d$ will be an effective method if:

a) the condition number, $\varkappa(C)$, is close to 1 so that the conjugate gradient algorithm converges rapidly, and

b) the multiplication $C \cdot y$ for any vector y can be computed efficiently and sparsely.

We shall examine condition (a) first. Note that

$$A = M - N$$

implies

$$C = M^{-1/2}AM^{-1/2} = I - (M^{-1/2}NM^{-1/2}) = I - R.$$

Hence

$$\varkappa(C) = \frac{\lambda_{max}(I-R)}{\lambda_{min}(I-R)} \geqslant 1$$

This condition number, $\varkappa(C)$, will be close to 1 if $\lambda_{max}(R)$ is small, which is true if all of the elements of R are small. (The condition number is 1 exactly when R is 0, or when $M = A$). We want, then, to choose M so that M represents A as well as possible (makes R as small as possible), and such that the choice for M allows for condition (b). We discuss a specific choice for M in section 3, the discussion of the model problem. Other examples are given in Concus, Golub, and O'Leary [3], O'Leary [4], and Meijerink and van der Vorst [9].

We now turn to condition (b); with a formal derivation of the actual algorithm used as "Preconditioned conjugate gradients," we show that condition (b) is met whenever M is such that the linear equation

$$Mz = w$$

can be solved efficiently and sparsely. Reconsider the conjugate gradient iteration to solve $Cz = d$, given an initial guess z_1:

```
(CG_c): set $\bar{r}_1 = d - Cz_1$;
        set $\bar{p}_1 = \bar{r}_1$;
        for $i = 2, 3, 4 \ldots$
            $\alpha_i = \dfrac{\bar{r}_i^T\bar{r}_i}{\bar{p}_i^T C\bar{p}_i}$
```

$$z_{i+1} = z_i + \alpha_i \bar{p}_i$$

$$\bar{r}_{i+1} = \bar{r}_i - \alpha_i C \bar{p}_i$$

$$\beta_i = \frac{\bar{r}_{i+1}{}^T \bar{r}_{i+1}}{\bar{r}_i{}^T \bar{r}_i}$$

$$\bar{p}_{i+1} = \bar{r}_{i+1} + \beta_i \bar{p}_i$$

In the above, the residual vectors $\{\bar{r}_i\}$ and descent directions $\{\bar{p}_i\}$ have bars on them to denote that they pertain to the scaled problem $Cz = d$. Now view (CG_c) from the original space in which we solve $Ax = b$. Define

$$x_i = M^{-1/2} z_i,$$

$$r_i = b - Ax_i$$

Then

$$\bar{r}_i = d - Cz_i = M^{-1/2} r_i$$

Similarily, we can define formally

$$p_i \equiv M^{-1/2} \bar{p}_i$$

Then

$$\alpha_i = \frac{\bar{r}_i{}^T \bar{r}_i}{\bar{p}_i{}^T C \bar{p}_i} = \frac{r_i{}^T (M^{-1/2})^T (M^{-1/2}) r_i}{p_i^T (M^{1/2})^T C M^{1/2} p_i} = \frac{r_i{}^T (M^{-1} r_i)}{p_i{}^T A p_i}$$

Updating z_i corresponds to taking

$$x_{i+1} = x_i + \alpha_i M^{-1/2} \bar{p}_i = x_i + \alpha_i p_i$$

The corrected residual is

$$\begin{aligned} r_{i+1} &= M^{1/2} \bar{r}_{i+1} \\ &= M^{1/2} (\bar{r}_i - \alpha_i C \bar{p}_i) \\ &= M^{1/2} \bar{r}_i - \alpha_i A M^{-1/2} \bar{p}_i \\ &= r_i - \alpha_i A\, p_i. \end{aligned}$$

The Gram-Schmidt coefficient β_i can be computed as:

$$\beta_i = \frac{\bar{r}_{i+1}{}^T \bar{r}_{i+1}}{\bar{r}_i{}^T \bar{r}_i} = \frac{r_{i+1}{}^T (M^{-1} r_{i+1})}{r_i{}^T (M^{-1} r_i).}$$

The new direction becomes:

$$p_{i+1} = M^{-1/2} \bar{p}_{i+1} = M^{-1/2} (\bar{r}_{i+1} + \beta_i \bar{p}_i)$$

$$= M^{-1/2}(M^{-1/2}\, r_{i+1}) + \beta_i p_i$$

$$= M^{-1} r_{i+1} + \beta_i p_i .$$

The result of these manipulations is that we can now write a conjugate-gradient like iteration to solve the equations $Ax = b$, but whose convergence rate depends on

$$C = M^{-1/2} A M^{-1/2}, \text{ not } A.$$

PRECONDITIONED CONJUGATE-GRADIENTS (PCG): Given initial guess x_1: compute

```
                $v_1 = Ax_1$
                $r_1 = b - Ax_1 = b - v_1$
solve           $Mu_1 = r_1$
and set         $p_1 = u_1$
    For $i = 2, 3, 4, \ldots$
    compute     $v_i = Ap_i$;
    set         $\alpha_i = \frac{r_i^T u_i}{p_i^T v_i} = \frac{r_i^T M^{-1} r_i}{p_i^T A p_i}$
    update      $x_{i+1} = x_i + \alpha_i p_i$
                $r_{i+1} = r_i - \alpha_i v_i \; (= r_i - \alpha_i A p_i)$
    solve       $Mu_{i+1} = r_{i+1}$
    set         $\beta_i = \frac{r_{i+1}^T u_{i+1}}{r_i^T u_i} \left(= \frac{r_{i+1}^T M^{-1} r_{i+1}}{r_i^T M^{-1} r_i}\right)$
    finally,    $p_{i+1} = u_{i+1} + \beta_i p_i$
```

We introduced new vectors $\{u_i\}$ and $\{v_i\}$ in the algorithm above to emphasize the fact that the matrices A and M appear only one time during each iteration, and in very specific ways. The matrix A appears only in the formation of the product Ap_i (or Ax_i), an operation which can be done very efficiently for most representations of sparse matrices. The matrix M appears only implicitly; we must be able to solve the linear systems $Mu = r$ efficiently and in little storage, and this is the primary restriction on M. Note that the matrix square-roots, $M^{1/2}$, do not appear at all. The purpose of the matrix M is to *scale* or *pre-condition* the problem so that convergence takes place very quickly. The cost, of course, is that each iteration now requires the solution of a linear equation as well as the formation of a matrix-vector product.

3. A model problem—the pressure equation.

The problem to be discussed here arises in a model of buoyant convection (Rehm and Baum [6]). Specifically, at each time step in the solution of a mixed hyperbolic-elliptic system, we are required to solve:

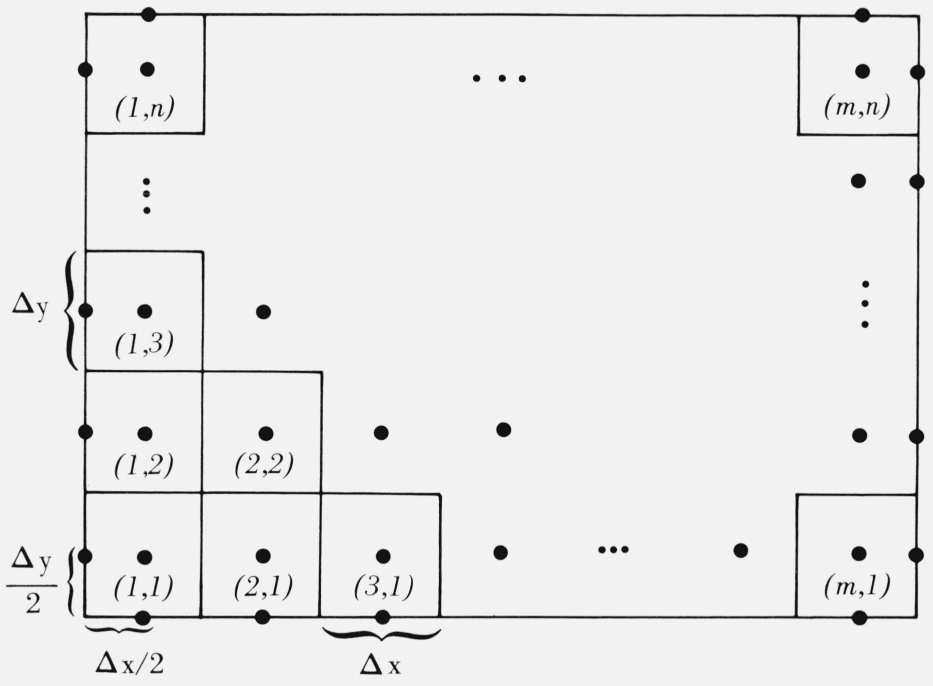
$$\nabla \cdot \left(\frac{1}{\varrho(x,y)} \nabla P(x,y) \right) = f(x,y) \qquad (20)$$

on a rectangle R subject to the following condition on the exterior normal derivative

$$\frac{\partial P}{\partial \eta}(x,y) = \varrho(x,y) \cdot g(x,y) \text{ on the boundary } \partial R.$$

Here ϱ denotes gas density and P the unknown pressure. We assume that ϱ depends on both spatial variables. Equation (1) then describes a *nonseparable* elliptic equation. Since we must solve this equation repeatedly, speed is paramount. Were this a separable equation, the discrete linear system discussed below could be solved directly by the cyclic reduction routines of Sweet and Schwarztrauber [10]. Were the density and its first and second derivatives known analytically, the problem could be converted to an ordinary Poisson equation by the techniques of Concus and Golub [2]. Neither of these conditions can be met, which forces us to consider the discrete linear system in more detail.

In the context of the buoyant convection problem, we are given the density ϱ, and g and f, only on a discrete set of points; the solution P will be produced on the same grid of discrete points. A grid suitable for the hydrodynamic calculations is:



The functions ϱ and f are given on the interior points; g is given at the circles on the boundary. Note that the interior grid is offset by one half grid spacing from the boundary.

The derivatives in (20) are replaced by second-order accurate centered finite differences. For example,

$$\frac{\partial}{\partial x} \cdot \left(\frac{1}{\varrho(x,y)} \frac{\partial}{\partial x} P(x,y) \right)$$

becomes

$$\frac{1}{\Delta x} \left(\frac{1}{\varrho(x + \frac{\Delta x}{2}, y)} \left[\frac{1}{\Delta x} (P(x + \Delta x, y) - P(x,y)) \right] - \right.$$

$$\left. \frac{1}{\varrho(x - \frac{\Delta x}{2}, y)} \left[\frac{1}{\Delta x} (P(x,y) - P(x - \Delta x, y)) \right] \right)$$

Note that we require the reciprocal of the density at an intermediate point: an $O(\Delta x)$ approximation will suffice to give second order accuracy for the operator. For consistency with the hydrodynamic equations in the buoyant convection model, we use the reciprocal of the average density:

$$\frac{1}{\varrho(x+\frac{\Delta x}{2}, y)} \cong \frac{2}{[\varrho(x+\Delta x, y)+\varrho(x,y)]}$$

Define $\varrho_{ij} = \varrho\,((i-1/2)\,\Delta x, (j-1/2)\,\Delta y)$, and similarly for P_{ij}, f_{ij}, and g_{ij}. Then the general equation for an interior grid point i, j not adjacent to the boundary is

$$\begin{aligned}
&\Bigg\{\Bigg[\underbrace{\left(\frac{-2}{\varrho_{i-1,j}+\varrho_{i,j}}\right)}_{d_1}+\underbrace{\left(\frac{-2}{\varrho_{i,j}+\varrho_{i+1,j}}\right)}_{d_2}\Bigg]\left(\frac{1}{\Delta x}\right)^2+\\
&\Bigg[\underbrace{\left(\frac{-2}{\varrho_{i,j-1}+\varrho_{i,j}}\right)}_{d_3}+\underbrace{\left(\frac{-2}{\varrho_{i,j}+\varrho_{i,j+1}}\right)}_{d_4}\Bigg]\left(\frac{1}{\Delta y}\right)^2\Bigg\}P_{ij}\\
&+\underbrace{\left(\frac{2}{\varrho_{i,j}+\varrho_{i-1,j}}\right)}_{-d_1}\left(\frac{1}{\Delta x}\right)^2 P_{i-1,j}\\
&+\underbrace{\left(\frac{2}{\varrho_{i,j}+\varrho_{i,j+1}}\right)}_{-d_2}\left(\frac{1}{\Delta x}\right)^2 P_{i+1,j} \qquad (21_{ij})\\
&+\underbrace{\left(\frac{2}{\varrho_{i,j-1}+\varrho_{i,j}}\right)}_{-d_3}\left(\frac{1}{\Delta y}\right)^2 P_{i,j-1}\\
&+\underbrace{\left(\frac{2}{\varrho_{i,j}+\varrho_{ij+1}}\right)}_{-d_4}\left(\frac{1}{\Delta y}\right)^2 P_{i,j+1} \qquad = f_{ij}
\end{aligned}$$

The adjustment for points adjacent to the boundary uses the second order centered approximation to the boundary derivative: on boundary k (left $=1$, right $=2$, lower $=3$, upper $=4$), the terms d_k are evaluated using the first interior mesh point and an image point (one-half grid spacing outside the region). Formally the terms d_1 are evaluated at $i = 1/2, j$ at the left boundary for example. The discretized form of the Neuman boundary conditions become

$$\left(\frac{1}{\Delta x}\right)^2\left(-\frac{1}{2}d_1p_{1,j}+\frac{1}{2}d_1p_{0,j}\right) = g_{1/2,j}/\Delta x \equiv g_1(j) \qquad (22_1)$$

$$\left(\frac{1}{\Delta x}\right)^2\left(-\frac{1}{2}d_2p_{m,j}+\frac{1}{2}d_2p_{m+1,j}\right) = g_{m+1/2,j}/\Delta x \equiv g_2(j) \qquad (22_2)$$

$$\left(\frac{1}{\Delta y}\right)^2\left(-\frac{1}{2}d_3p_{i,1}+\frac{1}{2}d_3p_{i,0}\right) = g_{i,1/2}/\Delta y \equiv g_3(i) \qquad (22_3)$$

$$\left(\frac{1}{\Delta y}\right)^2\left(-\frac{1}{2}d_4p_{i,n}+\frac{1}{2}d_4p_{i,n+1}\right) = g_{i,n+1/2}/\Delta y \equiv g_4(i) \qquad (22_4)$$

For a point adjacent to boundary k, subtract equation (22_k) from equation ($21_{i,j}$) which eliminates the d_k terms, and with them, all references to points i, j outside the grid. The right hand side is replaced by

$$f_{ij} - g_k$$

Corner points are treated by subtracting (22_k) and (22_ℓ) from (21_{ij}), where the corner is adjacent to both its k-th and ℓ-th boundary.

We arrange the grid points in the order
$\{(1,1), (2,1), \ldots, (m,1), (1,2), (2,2), \ldots, (m,2), \ldots, (1,n), \ldots, (m,n)\}$ and write the equation for each point in this order. The result is the matrix equation

$$\begin{bmatrix} T_1 & O_1 & & & & \\ O_1 & T_2 & O_2 & & & \\ & O_2 & T_3 & & & \\ & & & \ddots & & \\ & & & & T_{n-1} & O_{n-1} \\ & & & & O_{n-1} & T_n \end{bmatrix} \begin{bmatrix} \\ P \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \hat{f} \\ \\ \end{bmatrix}$$

where T_i is an $m \times m$ symmetric tridiagonal matrix, and O_i is an $m \times m$ diagonal matrix. We shall denote this matrix equation by

$$Ax = b \tag{23}$$

where b represents the right hand side, f, of the pressure equation, adjusted by the boundary data, g.

This is the linear algebraic system we need to solve. We make several observations:

1. A is symmetric
2. A is sparse. Only five diagonals contain non-zero elements.
3. The eigenvalues of A are all real and non-positive; A is a negative semi-definite matrix, generally of rank $mn - 1$. Hence $-A$ is a positive semi-definite matrix.

Although we could apply the method of conjugate gradients directly to $-A$, convergence would be very slow. O'Leary [4] and Meijerink and Van der Vorst [9] suggest several different approaches for creating scaling matrices M to use a preconditioned conjugate gradient scheme. One general approach is to consider that A originated from a differential equation and let M be the discrete operator from a separable differential equation which approximates the pressure equation.

A specific choice is suggested by Concus and Golub [2]: Let L be the discretization of the Poisson equation

$$\nabla^2 P = f$$

with Neumann boundary conditions (on the same staggered grid). Then

$$L = \begin{bmatrix} B_1 & c_1 I & & & & \\ c_1 I & B_2 & c_1 I & & & \\ & c_1 I & B_2 & \ddots & & \\ & & \ddots & \ddots & & \\ & & & & B_2 & c_1 I \\ & & & & c_1 I & B_1 \end{bmatrix} \tag{24}$$

where $$B_1 = \begin{bmatrix} a_1 & c_2 & & & & \bigcirc \\ c_2 & a_2 & c_2 & & & \\ & c_2 & a_2 & c_2 & & \\ & & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ \bigcirc & & & c_2 & a_2 & c_2 \\ & & & & c_2 & a_1 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} a_3 & c_2 & & & & \bigcirc \\ c_2 & a_4 & c_2 & & & \\ & c_2 & a_4 & c_2 & & \\ & & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ \bigcirc & & & c_2 & a_4 & c_2 \\ & & & & c_2 & a_3 \end{bmatrix}$$

with $a_1 = -\dfrac{1}{(\Delta x)^2} - \dfrac{1}{(\Delta y)^2}$

$$a_2 = -\frac{2}{(\Delta x)^2} - \frac{1}{(\Delta y)^2}$$

$$a_3 = -\frac{1}{(\Delta x)^2} - \frac{2}{(\Delta y)^2}$$

$$a_4 = -\frac{2}{(\Delta x)^2} - \frac{2}{(\Delta y)^2}$$

$$c_1 = \frac{1}{(\Delta y)^2}; \; c_2 = \frac{1}{(\Delta x)^2}$$

We assume that we have available a good subroutine for solving the equation

$$Lu = v,$$

e.g., subroutine BLKTRI from the NCAR package of subroutines for elliptic partial differential equations [10].

Let $D^{1/2}$ be the diagonal matrix with entries

$$d_{kk} = \sqrt{a_{kk}/l_{kk}},$$

the square root of the ratio of corresponding main diagonal entries of A and L. Then

$$M = D^{1/2} L D^{1/2}$$

is a symmetric negative semi-definite sparse matrix whose main diagonal is identical to that of A. Hence, in the splitting

$$A = M - N,$$

the matrix N has zero main diagonal and non-zero entries on at most four off-diagonals.

In the preconditioned conjugate gradient iteration we must be able to solve

$$Mz = r$$

This is done without excessive storage demands by:

a) compute $w = D^{-1/2} r$ (D is a diagonal matrix)
b) solve $Lu = w$ by cyclic reduction (BLKTRI)
c) compute $z = D^{-1/2} u$.

For computational convenience, to avoid the repeated multiplications by the diagonal matrix $D^{1/2}$ or its inverse, we replace the original problem

$$Ax = b$$

by

$$\hat{A}\hat{x} = \hat{b}$$

where

$$\hat{A} = D^{-1/2} A D^{-1/2}$$

$$\hat{b} = D^{-1/2} b$$

solve $Ax = b$ by preconditioned conjugate gradients with L as the scaling matrix, and finally compute

$$x = D^{-1/2} \hat{x}$$

This is the approach actually used in the computations reported in section 5.

4. The Preconditioned Conjugate Gradient Algorithm for a Semi-definite Matrix.

In our discussion of the discrete operators A and L in the previous section we have ignored one characteristic of these operators which renders them unsuitable for direct use in a conjugate gradient or preconditioned conjugate gradient iteration. The conjugate gradient algorithm requires that A be a positive-definite matrix; otherwise the function E may have zeroes other than at the solution of the linear equation. The algorithm becomes transparently a maximization algorithm for negative-definite matrices. (There is no need to change signs implicitly or explicitly to solve linear systems involving negative definite matrices, e.g., the discrete Laplacian with Dirichlet boundary conditions). However, our operators A and L are both negative semidefinite; they have negative eigenvalues, and also, one zero eigenvalue each.

In this section we will extend the theory of the conjugate gradient algorithm to allow for the special case of a semi-definite matrix with known nullspace. We shall then extend the preconditioned conjugate gradient algorithm to allow both A and M to be of this special type. The first part of the section will be elementary for readers well-versed in the conjugate gradient theory; the second part contains some new and unexpected results.

The problem is simple: A is a singular matrix. This is shown easily by noting that whenever a term d_k appears on the diagonal of A, the opposite term $-d_k$ appears on one of the off-diagonals. Hence, the sum of the coefficients in any row of A is exactly zero. But this is the same as saying

$$Ae = 0$$

where e is the vector $(1,1,1,\ldots 1)^T$. In general, $Ax = 0 \iff x = \alpha e$, where α is a real scalar.

The singularity of A is related to the Neumann boundary conditions for the pressure equation. If P is a solution to the differential equation so is $P + c$, where c is any constant. Similarly, if x is any solution to (3), so is $x + \alpha e$, where α is any scalar. But this is equivalent to adding the constant α to each point x_{ij} of the solution. The singularity of the operators also implies that not every system of equations has a solution. A system of equations is consistent if and only if it has a solution. The characterization of consistency for these operators is simple to express: The pressure equation is consistent $\iff$

$$\int\int_R f = \int_{\partial R} g \tag{25}$$

The linear equations $Ax = b$ are consistent $\iff$

$$\begin{aligned} &b^Te = 0, \text{ that is, when } b \text{ in Range}(A). \text{ But} \\ &b^Te = 0 \iff \sum_{i,j} b_{ij} = 0 \\ &\qquad\iff \sum_{i,j} f_{ij} = \sum_j (g_1(j) + g_2(j)) + \sum_i (g_3(i) + g_4(i)) \end{aligned}$$

the discrete analog of the integral equality (25)

Further, even when the equation is consistent, the solution is not unique. There is a unique solution of shortest length in the usual L_2 norm. For the continuous case it is the solution with mean pressure zero, i.e.,

$$\int\int_R P = 0.$$

In the discrete case the analog is

$$x^Te = 0;$$

again, the mean (discrete) pressure is zero.

We shall now discuss the modifications to the conjugate gradient algorithm which would be necessary to obtain this unique solution to a consistent system of equations. For generality, assume that we want to solve

$$Ax = b \tag{26}$$

where A is symmetric positive semi-definite of dimension k with known nullspace the span of $\{n\}$. Hence, the rank of A is one less than its dimension. Further, assume $b^Tn = 0$, so that (26) is a consistent system. (note: if (26) is not consistent, $\hat{b} = b - \dfrac{b^Tn}{n^Tn}$ satisfies the consistency condition, and any solution of $Ax = \hat{b}$ is a least squares solution of (26)).

Let the vectors $\{n, q_2, q_3, q_4, \ldots, q_k\}$ form an orthonormal basis for R^k. Then Range (A) = span $\{q_2, q_3, \ldots, q_k\}$

Let Q be the orthogonal matrix ($Q^T = Q^{-1}$)

$$Q = [n \vdots q_2 \vdots q_3 \vdots \dots \vdots q_k].$$

Then

$$Ax = b \iff (Q^T A Q)(Q^T x) = Q^T b \tag{27}$$

But, by the consistency condition $b^T n = 0$

$$Q^T b = \begin{bmatrix} 0 \\ d_1 \\ d_2 \\ \cdot \\ \cdot \\ \cdot \\ d_{k-1} \end{bmatrix}$$

Similarily

$$Q^T A Q = \left.\underbrace{\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & B \end{array}\right]}_{k-1}\right\} k-1$$

Hence, (27) really takes the form

$$\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & B \end{array}\right] \begin{bmatrix} w \\ z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_{k-1} \end{bmatrix} = \begin{bmatrix} 0 \\ d_1 \\ d_2 \\ \cdot \\ \cdot \\ \cdot \\ d_{k-1} \end{bmatrix} \tag{28}$$

Clearly w is arbitrary in (28). The equation holds whenever

$$Bz = d \tag{29}$$

a $(k-1)$ – dimensional problem. All solutions of (26) are given by

$$x = Q\left[\begin{array}{c} w \\ \hline z \end{array}\right]$$

where z solves (29). The unique minimal length solution is given by

$$x^+ = Q\left[\begin{array}{c} 0 \\ \hline z \end{array}\right]$$

By the assumptions on A, the matrix B is positive-definite and the conjugate gradient iteration can be used to solve (29). The condition number

$$\varkappa(B) = \frac{\lambda_{k-1}(B)}{\lambda_1(B)} = \frac{\lambda_k(A)}{\lambda_2(A)},$$

(where we assume the eigenvalues are ordered algebraically) governs the convergence of the algorithm.

We now show that we can solve the consistent linear system (26) directly by the conjugate gradient algorithm, with convergence rate determined by $\varkappa(B)$. The proof is elementary: we show that the algorithm, when started with an initial vector x_1 and a right hand side b which lie in the same invariant subspace of A, solves the problem while remaining entirely in that subspace.

Suppose we have the equation

$$Ax = b$$

with $b^T n = 0$ and $x_1{}^T n = 0$. Then the conjugate gradient algorithm, when applied to A, produces vectors x_i, r_i and p_i exactly equal to

$$x_i = Q\left[\begin{array}{c} 0 \\ \hdashline z_i \end{array}\right]$$

$$r_i = Q\left[\begin{array}{c} 0 \\ \hdashline \hat{r}_i \end{array}\right]$$

and

$$p_i = Q\left[\begin{array}{c} 0 \\ \hdashline \hat{p}_i \end{array}\right]$$

where z_i, $\hat{r}_i$ and $\hat{p}_i$ are the vectors produced by the conjugate gradient algorithm for

$$Bz = d$$

with initial vector $z_1 =$ the last $k-1$ entries of $Q^T x_1$. We note that

$$Q^T r_1 = Q^T(b - Ax_1) = Q^T b - Q^T A(QQ^T)x_1 = \left[\begin{array}{c} 0 \\ \hdashline d \end{array}\right] - \left[\begin{array}{c} 0 \\ \hdashline Bz_1 \end{array}\right] = \left[\begin{array}{c} 0 \\ \hdashline \hat{r}_1 \end{array}\right],$$

where it is essential that $b^T n = 0$. It follows that

$$Q^T p_1 = \left[\begin{array}{c} 0 \\ \hdashline \hat{p}_1 \end{array}\right].$$

So assume the required properties hold for x_i, r_i and p_i. We shall show the iteration preserves these properties for x_{i+1}, r_{i+1}, and p_{i+1}. The first step in the iteration is to compute

$$\alpha_i = \frac{r_i{}^T r_i}{p_i{}^T A p_i},$$

which by induction is

$$\alpha_i = \frac{\left[\begin{array}{c} 0 \\ \hdashline \hat{r}_i \end{array}\right]^T Q^T Q \left[\begin{array}{c} 0 \\ \hdashline \hat{r}_i \end{array}\right]}{\left[\begin{array}{c} 0 \\ \hdashline \hat{p}_i \end{array}\right]^T Q^T A Q \left[\begin{array}{c} 0 \\ \hdashline \hat{p}_i \end{array}\right]}$$

$$= \frac{\hat{r}_i{}^T \hat{r}_i}{\hat{p}_i{}^T B \hat{p}_i},$$

since $Q^TQ = I$ and

$$Q^TAQ = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & B \end{array}\right]$$

Hence, $\hat{\alpha}_i = \alpha_i$, and it follows that

$$x_{i+1} = x_i + \alpha_i p_i = Q\left[\begin{array}{c} 0 \\ \hline z_i \end{array}\right] + \alpha_i Q\left[\begin{array}{c} 0 \\ \hline \hat{p}_i \end{array}\right] = Q\left[\begin{array}{c} 0 \\ \hline z_i + \hat{\alpha}_i p_i \end{array}\right] = Q\left[\begin{array}{c} 0 \\ \hline z_{i+1} \end{array}\right]$$

Then

$$\begin{aligned} r_{i+1} &= r_i - \alpha_i A p_i \\ &= Q\left[\begin{array}{c} 0 \\ \hline \hat{r}_i \end{array}\right] - \alpha_i AQ\left[\begin{array}{c} 0 \\ \hline \hat{p}_i \end{array}\right] \\ &= Q\left[\begin{array}{c} 0 \\ \hline \hat{r}_i \end{array}\right] - \alpha_i Q(Q^TAQ)\left[\begin{array}{c} 0 \\ \hline \hat{p}_i \end{array}\right] \\ &= Q\left[\begin{array}{c} 0 \\ \hline \hat{r}_i - \hat{\alpha}_i B p_i \end{array}\right] = Q\left[\begin{array}{c} 0 \\ \hline \hat{r}_{i+1} \end{array}\right] \end{aligned}$$

That $\hat{\beta}_i = \beta_i$ follows from exactly the same reasoning that showed equality for the numerators of α_i and $\hat{\alpha}_i$. It is equally easy to show that

$$p_{i+1} = r_{i+1} + \beta_i p_i \quad <=> \quad Q^T p_{i+1} = \left[\begin{array}{c} 0 \\ \hline \hat{r}_{i+1} + \hat{\beta}_i \hat{p}_i \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \hat{p}_{i+1} \end{array}\right].$$

Thus the conjugate gradient iteration will succeed in solving a consistent system.

We should note that the condition that $x_1^T n = 0$ is required only to assure that the solution also satisfies this property. If the minimal length solution is not desired, this condition can be ignored. The condition that $b^T n = 0$, that $Ax = b$ be consistent, is essential. If the system is not consistent, the initial residual r_i, and all successive residuals, will have the same component in the direction of the nullspace. Suppose that b (and r_1 and p_1) has a component $\gamma \cdot n$ of the nullspace. The recursion for $\{r_i\}$

$$r_{i+1} = r_i - a_i A p_i$$

shows that r_{i+1} has a component $\gamma \cdot n$ of the nullspace for all i. This, of course, also follows from the property that r_{i+1} is a residual vector for Ax_i and b. The directions $\{p_i\}$, and the approximate solutions $\{x_i\}$, have components of n which increase with i (in fact, rapidly).

It suffices to examine the recursion for p_i

$$p_{i+1} = r_{i+1} + \beta_i p_i$$

where $\beta_i > 0$. If δ_i is the component of n in the vector p_i,

$$\delta_{i+1} = (\gamma + \beta_i \delta_i) = (\gamma + \beta_i(\gamma + \beta_{i-1} \cdot \delta_{i-1})) = \dots$$

Thus the direction's component of the nullspace increases. At the same time, p_{i+1} is shorter than r_{i+1} in the A norm, so the relative component in the direction of the nullspace grows. In the limit, the directions may become nearly parallel (they are still A-orthogonal, but A does not induce a metric). We can see this also by looking at the limiting case when a iteration is started with an actual solution vector. The residual vector (which is not a descent direction) and the initial direction, are in the nullspace—the initial step α_1 is infinite.

The last example given above implies that the convergence rate for obtaining the least squares solution to an inconsistent system no longer depends on $\varkappa(B)$. The convergence obtained in practice may be much slower than the bound obtainable for a consistent system. In practice, we are interested only in minimizing the residual, which we can ensure by working with a consistent system. In actual finite precision computing, the residuals and directions may wander slightly into the nullspace, in which case we are in the same situation as if we started with a slightly inconsistent system. The effects will be negligible unless the rate of convergence is very slow. In any case, the effects may be suppressed by explictly reorthogonalizing the directions p_i to the nullspace.

In the preconditioned conjugate gradient algorithm we replace the system

$$Ax = b$$

by

$$Cz = d$$

where

$$C = VAV,$$

V some positive definite symmetric matrix. In the special case of a semi-definite A, a natural choice for a scaling matrix may also be semi-definite. This is the case for the two operators discussed in section 3. We have two possible cases to consider: V positive definite and V semi-definite.

The case of a positive definite V is little changed from the ordinary semi-definite conjugate gradient algorithm. We take $M = (V^2)^{-1}$, or more directly, $V = M^{-1/2}$. The rank of the matrix C is $k-1$ and its nullspace is

$$\text{span}\,\{M^{1/2} n_A\}$$

where n_A denotes a non-zero vector in the nullspace of A. The convergence condition for C required that $\bar{r}_i$ and $\bar{p}_i$ be orthogonal to $M^{1/2} n_A$.
The former condition will hold if $Ax = b$ is consistent since then

$$\bar{r}_i = M^{-1/2}(b - Ax_i)$$

and

$$(b - Ax_i)^T n_A = 0$$

The latter condition is equivalent to

$$p_i \perp M n_A$$

since

$$p_i = M^{-1/2} \bar{p}_i$$

But clearly $p_i = M^{-1} r_1$ is orthogonal to $M\, n_A$, since $r_1{}^T n_A = 0$. By induction, if $p_i{}^T M n_A = 0$,

$$p_{i+1}{}^T M n_A = (r_{i+1}{}^T M^{-1})\, M n_A + \beta_i\, (p_i{}^T M n_A) = r_{i+1}{}^T n_A = 0.$$

Thus, convergence is governed by the pseudo-condition number $\dfrac{\lambda_\varkappa(C)}{\lambda_2(C)}$, the residual vectors r_i remain those of a consistent system, and the coefficients α_i and β_i are determined by an iteration remaining in a subspace of the range of C. However, the scaled directions p_i are allowed to venture into the nullspace of A, and hence, the solution vector x is no longer necessarily the minimal length solution. This may be repaired easily at the end of the iteration by computing

$$x^+ = x - \frac{x^T n_A}{n_A{}^T n_A} \; n_A .$$

The other natural case is one in which the scaling matrix M is also positive semi-definite, with the null-space generated by a known vector n_M. The linear operator corresponding to M^{-1} in the definite case will be M^+, the pseudo-inverse of M. (For our purposes, we will need only a method for solving consistent systems $Mz = w$. Our knowledge of the nullspace of M then enables us to compute the minimal length least squares solution, M^+u, for any right hand side u). Formally, the matrix V appearing in the definition of C is $(M^+)^{1/2}$, so

$$C = (M^+)^{1/2} A (M^+)^{1/2} \cdot \tag{6}$$

Several conditions which must be met by M are immediate. Since nullspace $(C) \supseteq$ nullspace (M), the rank of M must be at least the rank of A. For our specific problem rank (M) must be at least $k-1$. Otherwise C has rank at most $k-2$ and cannot possibly span the entire $k-1$ dimensional space of possible solutions to $Ax = b$. To guarantee that C has rank $k-1$, we must have rank $(M) \geqslant k - 1$ and also that the nullspace of M is not orthogonal to the nullspace of A. If the latter condition fails, we know that n_A lies in the range of M (since it is orthogonal to the orthogonal complement of the range of M). Hence, by symmetry of M, n_A lies in the range of $(M^+)^{1/2}$. There exists a vector z such that $(M^+)^{1/2} z = n_A$ and $z \perp n_M$. We would have:

$$C\, n_M = (M^+)^{1/2} A\, ((M^+)^{1/2} n_M) = (M^+)^{1/2} A\, 0 = 0$$

$$Cz = (M^+)^{1/2} A\, ((M^+)^{1/2} z) = (M^+)^{1/2} A n_A = (M^+)^{1/2}\, 0 = 0$$

Thus, the dimension of the nullspace of C would be at least two.

Recall now the motivation for the preconditioned algorithm. The rate of convergence of the conjugate gradient algorithm is determined by the condition number, $\varkappa(A)$, which is large for most discrete elliptic operators. To accelerate the coverage of the conjugate gradient algorithm, we replaced A by the preconditioned operator $C = M^{-1/2} A M^{-1/2}$, where we presume that $\varkappa(C) \approx 1$. Our presumption can only be true if $\varkappa(A) \approx \varkappa(M)$, i.e., the preconditioning matrix must have roughly the same poor conditioning as has A. This requirement follows from the inequality

$$\varkappa(C) \geqslant \max \left\{ \frac{\varkappa(A)}{\varkappa(M^{-1})}, \frac{\varkappa(M^{-1})}{\varkappa(A)} \right\}$$

where we note that $\varkappa(M^{-1}) = \varkappa(M)$. A brief proof of this inequality is given below:

$$\varkappa(C) = \lambda_{max}(C) \,/\, \lambda_{min}(C).$$

However, the eigenvalues of C are the same as the eigenvalues of $M^{-1} A$. We can bound the eigenvalues of this latter product as:

$$\lambda_{max}(M^{-1} A) \geqslant \lambda_{max}(M^{-1}) \cdot \lambda_{min}(A),$$

$$\lambda_{min}(M^{-1} A) \leqslant \lambda_{min}(M^{-1}) \cdot \lambda_{max}(A).$$

It follows that

$$\varkappa(C) \geqslant \frac{\lambda_{min}(\mathrm{A})}{\lambda_{max}(\mathrm{A})} \cdot \frac{\lambda_{max}(\mathrm{M})}{\lambda_{min}(\mathrm{M})}$$

$$= \varkappa(M) / \varkappa(A).$$

The other inequality above follows by a similar argument.

We can obtain suitable preconditioning matrices for most positive definite discrete elliptic operators only by using poorly conditioned matrices. The required poor conditioning of M may be even worse in the case where both A and M are semi-definite. The preconditioning breaks down if

$$n_A{}^T n_M = 0.$$

We shall now show that the preconditioning nearly breaks down, in the sense that M must be very ill-conditioned, if the nullspaces are almost orthogonal. Assume that n_A and n_M have length one, and that

$$n_A{}^T n_M = \delta,$$

which is much less than one. We can write A and M^+ in their respective eigendecompositions

$$A = U\, D_A\, U^T,$$

$$M^+ = V\, D_{M+}\, V^T,$$

where the zero eigenvalue of A and of M^+ is given first. Then

$$C = M^{+1/2} A\, M^{+1/2} = S\, S^T,$$

where S is defined by

$$S = V\, D_{M+}{}^{1/2}\, V^T U\, D_A{}^{1/2}.$$

For each of the matrices S, M^+, A, and X, define the condition number of the matrix as the condition number of the matrix restricted to the orthogonal complement of its nullspace. Then

$$\varkappa(S) = \varkappa[(D_{M+}{}^{1/2})(X)\,(D_A{}^{1/2})].$$

In the above equation, the matrix X is the $(k-1)$ by $(k-1)$ matrix obtained by removing the first row and column from $V^T U$, that is,

$$V^T U = \left[\begin{array}{c|c} \delta & a^T \\ \hline b & X \end{array}\right]$$

We now note that the inequality

$$\varkappa(BC) \geqslant \max\left\{ \frac{\varkappa(B)}{\varkappa(C)}, \frac{\varkappa(C)}{\varkappa(B)} \right\}$$

holds for unsymmetric matrices B and C. (The proof follows from the triangle inequality for matrix norms,

$$\|B\| \leqslant \|BC\|\ \|C^{-1}\|.)$$

When we extend this inequality to products of three matrices we obtain the relatively weak result that

$$\varkappa(BCD) \geqslant \max \left\{ \frac{\varkappa(B)}{\varkappa(C)\varkappa(D)}, \frac{\varkappa(C)}{\varkappa(B)\varkappa(D)}, \frac{\varkappa(D)}{\varkappa(B)\varkappa(C)} \right\}$$

In the context of our preconditioned operator, only $\varkappa(X)$ is unknown. However, Alan Cline [11] has shown that when $\delta \ll 1$,

$$\varkappa(X) \cong 1 / \delta.$$

Thus, the condition number of the preconditioned operator C is bounded below by:

$$\varkappa(C) \geqslant (1 / \delta)^2 / (\varkappa(A)\, \varkappa(M)).$$

Since $\varkappa(A)$ is fixed, this implies that $\varkappa(M)$ must be large whenever δ is small.

V. Numerical Results

A code, written in FORTRAN, was developed to implement the preconditioned conjugate gradients scheme discussed in previous sections. Care was taken in the preparation of the code to make it portable and to introduce many comments for clarity. The code has been run successfully under a variety of conditions and has been compared with analytical results to determine its accuracy. In this section a description is given of some of the computations used to determine the performance of this code.

The model problem, the pressure equation, was discussed in section III. Special cases of this general non-separable elliptic equation were used to test the code for accuracy. All production runs of this code were performed when the code was imbedded in a larger linear or nonlinear fluid dynamics computation; timing studies were performed in such an environment.

Within the code, two tests are used to terminate the conjugate gradient iteration; these depend upon two specified parameters, the maximum number of iterations (less than or equal to 50) and a maximum relative residual, ε. (The relative residual is defined as the norm of the residual divided by the norm of the right hand side in the scaled problem discussed in sections 2 and 4.) In all successful computations to date, the iteration is terminated after a relatively small number of iterations by the relative residual norm satisfying the criterion that it be below ε.

To test the code under the simplest conditions, a Poisson equation on the unit square was discretized and solved with homogeneous Neumann conditions applied at the boundary. In continuous form this problem is

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \tilde{f}(x,y) \equiv \cos(\varkappa\pi\, x)\cos(\ell\pi\, y)$$

on $0 \leqslant x \leqslant 1$ and $0 \leqslant y \leqslant 1$. The boundary conditions are

$$\frac{\partial p}{\partial x} = 0 \text{ at } x = 0 \text{ and } x = 1,$$

$$\frac{\partial p}{\partial y} = 0 \text{ at } y = 0 \text{ and } y = 1.$$

The solution to this problem is

$$p(x,y) = -\frac{1}{(\ell\pi)^2 + (\varkappa\pi)^2} \cos(\varkappa\pi\, x)\cos(\ell\pi\, y)$$

When this problem is discretized to second order accuracy on the "staggered grid" (a grid displaced one half incremental unit δx in the x-direction and one half incremental unit δy in the y-direction) as discussed in section III, it can be written

$$\frac{1}{\delta x^2}(\bar{p}_{i+1,j} - 2\bar{p}_{ij} + \bar{p}_{i-1,j}) + \frac{1}{\delta y^2}(\bar{p}_{i,j+1} - 2\bar{p}_{ij} + \bar{p}_{i,j-1}) = \bar{f}_{ij}$$

Where

$$\bar{f}_{ij} = \cos[\frac{\varkappa\pi}{m}(i - ½)] \cos[\frac{\ell\pi}{n}(j - ½)]$$

for

$$1 \leqslant i \leqslant m \text{ and } 1 \leqslant j \leqslant n.$$

Here δx is the mesh spacing in the x-direction, $\delta x = 1/m$ (m is the number of mesh cells in the x-direction) and δy is the mesh spacing in the y-direction, $\delta y = 1/n$ (n is the number of cells in the y-direction). The boundary conditions are

$$\bar{p}_{0,j} = \bar{p}_{1,j} \text{ and } \bar{p}_{m+1,j} = \bar{p}_{m,j} \text{ for } 1 \leqslant j \leqslant n$$

$$\bar{p}_{i,0} = \bar{p}_{i,1} \text{ and } \bar{p}_{i,n+1} = \bar{p}_{i,n} \text{ for } 1 \leqslant i \leqslant m$$

The solution to this linear algebraic system is

$$\bar{p}_{ij} = -\frac{1}{\left(2m \sin\frac{\varkappa\pi}{2m}\right)^2 + \left(2n \sin\frac{l\pi}{2n}\right)^2} \cos[\frac{\varkappa\pi}{m}(i-½)] \cos[\frac{\ell\pi}{n}(j-½)]$$

$$\text{for } 0 \leqslant i \leqslant m+1,\ 0 \leqslant j \leqslant n+1$$

The solution to this simple discretized Poisson equation was computed using the code, and the solution compared with the analytical solution given above.

A small test problem, $m = n = 7$, was used to determine the accuracy obtainable when $\varepsilon = 10^{-6}$. Comparison with the exact solution demonstrated that the components of the solution vector obtained from the computation agreed to at least six significant figures with the exact solution. Generally the agreement was much better, being seven or eight significant figures for most values of i and j. (Note that the UNIVAC 1108, on which all computations were run, carries about eight significant figures.)

As a larger test problem, the equations with $m = n = 31$ were solved with $\varepsilon = 10^{-6}$. For this computation agreement was obtained to a few parts in the sixth significant figure.

A second test problem, a discretized approximation to a separable elliptic equation, was also solved analytically and using the code. Comparison of these results indicated that the accuracy was similar to that obtained in the first test problem.

The code was then imbedded in a linear finite difference computation obtained from a second-order discretization of a set of equations arising in fluid dynamics. These equations describe two-dimensional internal gravity waves in a stratified ambient fluid within a rectangular enclosure. The interest in this problem is discussed, the continuous and discrete equations are presented and exact analytical solution to the continuous and discrete problem are given by Baum and Rehm [8]. Additional linear computations using this code on a somewhat more general fluid-flow problem are described in Rehm and Baum [7]. In this latter paper the more general linear finite difference equations are given, and the computational procedure for solving them is presented. The manner in which the preconditioned conjugate gradients code is used in solving the elliptic pressure equation is discussed. In all of these linear computations, the equation for the pressure is separable; it is only in the general, nonlinear computations that the equation for the pressure is nonseparable.

For the linear computations reported in these papers, the number of iterations required to obtain a relative residual error less than $\varepsilon = 10^{-5}$ or 10^{-6} generally varied between 2 and 4. A large number of such computations have been run.

A representative one is a calculation for which $m = 15$, $n = 16$ and $\varepsilon = 10^{-6}$; in this computation the pressure-solver was called 200 times. Two iterations to convergence were taken ten of the first eleven times it was called, each call using about 0.5 s of CPU time on the NBS Univac 1108. Thereafter, each subsequent call required only one iteration to convergence taking about 0.35 s of CPU. An indication of the rate of covergence of the algorithm is obtained by taking

$$\delta \equiv \frac{\log_{10}(\text{Initial relative residual}) - \log_{10}(\text{final relative residual})}{\text{number of iterations}}$$

For the computation reported above, this rate of convergence was about 4.5 when two iterations were taken and about 5.5 when one iteration was taken.

It should be noted that these computations determine a flow field which evolves with time. Except for the first time step, the pressure vector at the previous time step is used as the initial guess for the pressure vector each time the elliptic-solver is called. Hence the guess at each calling is quite good and convergence is rapid.

Computations of a set of nonlinear finite difference equations generalizing the linear ones described above have also been run. As noted previously, in this case the elliptic equation for the pressure is nonseparable. In a limited number of computations, the PCG code has been found to perform well in this case also. A representative computation, one for which $I = 31, J = 31$ and $\varepsilon = 10^{-5}$, was found to take between 2 and 5 iterations for convergence at each call. The CPU time taken was slightly over 2 s for 2 iterations and slightly under 5 s for 5 iterations (very roughly a second per iteration generally). In this calculation δ defined above was found to vary between about one and two.

In the Appendix to [15] a listing of the elliptic-solver, called FASTSL, is given. To use this code, subroutines from EISPACK, the Argonne Code Center package, and BLKTRI, a subroutine in the NCAR package developed by Schwarztrauber and Sweet[10] are required.

The authors wish to acknowledge the help of our colleagues, Mr. Martin Cordes of the Center for Applied Mathematics (C.A.M.) and Dr. Howard Baum of the Center for Fire Research (C.F.R.). This study would not have been completed without their help. We also wish to thank Drs. Dianne O'Leary and Francis Sullivan for reading and correcting the manuscript. One of us (J. L.) would like to express his appreciation to Drs. B. H. Colvin and F. C. Johnson for providing the opportunity to undertake this study in the C.A.M. during the summer of 1977.

VI. References

[1] Rice, John, Algorithm Progress in Solving Partial Differential Equations, SIGNUM Newsletter, Vol. **11**, No. 4, Dec. 1976.

[2] Concus, Paul and Golub, Gene H., Use of fast direct methods for the efficient numerical solution of non separable elliptic equations, SIAM J. Numer. Anal., **10**, 6 (1973).

[3] Concus, Paul; Golub, Gene H. and O'Leary, Dianne P., A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations, in *Sparse Matrix Computations*, J. Bunch and D. Rose, ed. (Academic Press, New York, 1975).

[4] O'Leary, Dianne P., Hybrid Conjugate Gradient Algorithms, Thesis, Computer Science Department, Stanford University, 1976, TR STANS–CS 548.

[5] O'Leary, Dianne P. and Widlund, Olof, Capacitance Matrix Methods for the Helmholtz Equation on General Three Dimensional Regions, Math of Comp. **33**, No. 147, 849–879 (July 1979).

[6] Rehm, Ronald G. and Baum, Howard R., The Equations of Motion for Thermally Driven, Buoyant Flows, J. Res. Nat. Bur. Std. (U.S.), **83**, No. 3, 297–308 (May–June 1978).

[7] Rehm, Ronald G.; Baum, Howard R.; Lewis, John G.; Cordes, Martin R.; A Linearized Finite—Difference Computation of Fluid Heating in an Enclosure, NBSIR 79–1754, May 3, 1979.

[8] Baum, Howard R. and Rehm, Ronald G., Finite Difference Solutions for Internal Waves in Enclosures, N.B.S. Internal Report, in preparation.

[9] Meijerink, J. A. and van der Vorst, H. A. An Iterative Solution for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix, Math. of Comp., **31**, 137, (Jan. 1977).

[10] Schwarztrauber, Paul and Sweet, Roland, Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations, NCAR Technical Note IA-109, July 1975.

[11] Cline, A., Private Communication.

[12] Reid, J. K., On the Method of Conjugate Gradients for the Solution of Large Sparse Systems of Linear Equations, in *Large Sparse Sets of Linear Equations,* J. K. Reid ed., (Academic Press, New York, 1971), 231-254.

[13] Hestenes, M. R. and Stiefel, E., Methods of Conjugate Gradients for Solving Linear Systems, J. Res. Nat. Bur. Std. (U.S.), **49**, No. 6, 409-436 (Dec. 1952).

[14] Daniel, J. W., The cg method for linear and nonlinear operator equations, SIAM J. Numer. Anal. **4** (1967) 10-26.

[15] Lewis, J. G. and Rehm, R. G., The Numerical Solution of a Nonseparable Elliptic Partial Differential Equation by Preconditioned Conjugate Gradients, NBSIR80-2056.