Pergamon

Computers Math. Applic. Vol. 30, No. 11, pp. 25–37, 1995

0898-1221/95 $9.50 + 0.00

0898-1221(95)00161-1

A Constrained Conjugate Gradient Method and the Solution of Linear Equations

M. H. B. M. SHARIFF
School of Computing and Mathematics
University of Teesside
Middlesbrough, Cleveland TS1 3BA, U.K.

(Received May 1995; accepted June 1995)

Abstract—A conjugate gradient method for solving minimization problems subject to linear equality constraints is developed. The method can be considered as an extension of the unconstrained Fletcher and Reeves algorithm. Proofs of several theorems related to the method are given. An application of the method is to provide an alternative method to solve a real system of linear equations for varying right-hand sides. Numerical solutions can be obtained in one iteration which leads to the development of a new direct method to invert a square matrix. Since this direct method involves matrix-vector multiplications and outer products which are easy to parallelize, it has an advantage over existing well-known direct methods. Numerical examples are given.

Keywords—Constrained conjugate gradient, Matrix inversion.

1. INTRODUCTION

In this paper, a linear equality constrained conjugate gradient method is developed which is used to provide an alternative iterative method to solve a real system of linear equations,

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

and help to prove a new direct method algorithm to obtain $\mathbf{A}^{-1}$. Here, $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{x}^{\mathrm{T}} = (x_1, x_2, \dots, x_n)$ is the unknown and $\mathbf{b}$ is a given right-hand side vector. It is assumed that the system of equations given above has a unique solution. The matrix $\mathbf{A}$ is not assumed to have special features, i.e., it is not assumed to be symmetric positive definite, sparse, etc. It is noted, however, that if $\mathbf{A}$ has special features the proposed methods can exploit them to achieve efficiency. Our iterative method is based on the proposed conjugate gradient method with linear equality constraints. We note that this is not the variable metric methods [1–4] that are also known as the conjugate directions method; we may, of course, use the variable metric methods to solve problem (1) based on the concept described in Section 3. The proposed method can be regarded as an extension of Fletcher and Reeves method [5] to linear equality constraints and can be easily extended to take account linear inequality constraints. However, for the purpose of solving problem (1), we only consider equality constraints.

In order to have confidence in the conjugate gradient algorithm, which on the surface looks not much different from the unconstrained conjugate gradient of Fletcher and Reeves [5] and from the algorithm developed by Arora and Li [6], the proofs of several theorems are given.

In Section 3, using the orthogonal projection matrix $\mathbf{H}_m$, the algorithm is adapted to solve problem (1). Faster convergence can be obtained by introducing a scalar ε (see Section 3 for more details) in our formulation. In the numerical examples, we show that with judicious use

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

of scalar ε, solutions are obtained in one iteration only. We also prove analytically and in exact arithmetic that as $\varepsilon \to 0$ solution of (1) can be obtained in exactly one iteration, and hence, leads to development of a new direct method to obtain $\mathbf{A}^{-1}$.

It is common knowledge that the use of iterative techniques gives up the ability to easily re-solve (1) for new right-hand sides. However, in our method, $\mathbf{H}_m$ is fixed for fixed $\mathbf{A}$, and solutions can be obtained in one iteration, hence, solving several systems with a fixed matrix $\mathbf{A}$ and varying vectors $\mathbf{b}$ is but marginally more expensive than solving just a single system.

Results and numerical discussions are presented in Section 4.

2. CONJUGATE GRADIENT METHOD WITH LINEAR EQUALITY CONSTRAINTS

We consider the problem

$$\text{Minimize } f = \frac{\mathbf{x}^{\mathsf{T}}\mathbf{G}\mathbf{x}}{2} - \mathbf{w}^{\mathsf{T}}\mathbf{x} \tag{2}$$

$$\text{Subject to } \mathbf{C}^{\mathsf{T}}\mathbf{x} = \mathbf{h} \tag{3}$$

where $\mathbf{G}$ is a positive definite symmetric matrix, $\mathbf{w}$ is a given vector, $\mathbf{C}$ is an $n \times m$ matrix, $\mathbf{h}$ is a given $m \times 1$ vector and $m < n$.

It is assumed that the columns of $\mathbf{C}$ are linearly independent. These constraints restrict the solution to m hyperplanes. The intersections of m linearly independent hyperplanes is, in general, an affine subspace (or flat) of $\mathbb{R}^n$ and will be denoted by $\mathbb{F}_m$.

2.1. Proposed Algorithm

The heart of the algorithm is to use a projection matrix $\mathbf{H}$, obtained below, so that the directions of search will always be in the feasible region.

The imposition of m linearly independent equality constraints on an n-dimensional problem reduces the dimensionality of the optimization to $n - m$. We note that any n-vector $\mathbf{x}$ has a unique expansion as a linear combination of the columns of $\mathbf{W}$ and $\mathbf{Z}$, i.e.,

$$\mathbf{x} = \mathbf{Z}\mathbf{y} + \mathbf{W}\mathbf{q},$$

for some $n - m$ vector $\mathbf{y}$ and m vector $\mathbf{q}$. $\mathbf{W}$ denotes any matrix whose columns form a basis for the range space of $\mathbf{C}$, the columns of $\mathbf{Z}$ form a basis for the null space of $\mathbf{C}^{\mathsf{T}}$ and $\mathbf{Z}$ and $\mathbf{W}$ define complimentary subspaces. For feasible $\mathbf{x}$, we have

$$\mathbf{C}^{\mathsf{T}}\mathbf{x} = \mathbf{C}^{\mathsf{T}}(\mathbf{Z}\mathbf{y} + \mathbf{W}\mathbf{q}) = \mathbf{b}.$$

Since $\mathbf{C}^{\mathsf{T}}\mathbf{Z} = \mathbf{0}$, if follows that

$$\mathbf{C}^{\mathsf{T}}\mathbf{W}\mathbf{q} = \mathbf{b}. \tag{4}$$

By definition of $\mathbf{W}$, the matrix $\mathbf{C}^{\mathsf{T}}\mathbf{W}$ is nonsingular and thus from (4), $\mathbf{q} = \mathbf{q}^*$ is uniquely determined. Hence, the transformation for feasible $\mathbf{x}$, i.e.,

$$\mathbf{x} = \mathbf{Z}\mathbf{y} + \mathbf{W}\mathbf{q}^* \tag{5}$$

can be considered as elimination of constraints by transformation of variables. Substituting (5) in (2), we have

$$f(\mathbf{x}) = f(\mathbf{Z}\mathbf{y} + \mathbf{W}\mathbf{q}^*).$$

Any feasible direction can be written as

$$\mathbf{p} = \mathbf{Z}\mathbf{v},$$

where $\mathbf{v}$ is any $n - m$ vector of changes in $\mathbf{y}$. The reduced gradient $\frac{\partial f}{\partial \mathbf{y}}$ is $\mathbf{Z}^\top \mathbf{g}$ (where $\mathbf{g} = \frac{\partial f}{\mathbf{x}}$), and so $\mathbf{y}$ should be changed so that

$$\mathbf{v} = -\mathbf{Z}^\top \mathbf{g}.$$

Hence,

$$\mathbf{p} = -\mathbf{Z}\mathbf{Z}^\top \mathbf{g}.$$

We note that the projection matrix $\mathbf{H} = \mathbf{Z}\mathbf{Z}^\top$ is not unique and in general does not project into itself, i.e., $\mathbf{H}\mathbf{H} \neq \mathbf{H}$. The orthogonal projection matrix $\mathbf{H}_m$ is related to $\mathbf{Z}$ by the relation

$$\mathbf{H}_m = \mathbf{Z}\left(\mathbf{Z}^\top \mathbf{Z}\right)^{-1} \mathbf{Z}^\top.$$

If $\mathbf{Z}^\top \mathbf{Z} = \mathbf{I}$ (the identity matrix), we have

$$\mathbf{p} = -\mathbf{H}_m \mathbf{g}.$$

Our algorithm takes the form

```
initialise
        $\mathbf{x} = \mathbf{x}_0$ (which satisfies (3))
        $\mathbf{g}_0 = \mathbf{G}\mathbf{x}_0 - \mathbf{w}$
        $\mathbf{z}_0 = \mathbf{H}\mathbf{g}_0$
        $\mathbf{d}_0 = -\mathbf{z}_0$
for  $i = 0, 1, 2, \ldots$
        $\alpha_i = \frac{-\mathbf{g}_i^\top \mathbf{d}_i}{\mathbf{d}_i^\top \mathbf{G}\mathbf{d}_i}$        (6)
        $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$        (7)
        $\mathbf{g}_{i+1} = \mathbf{g}_i + \alpha_i \mathbf{G}\mathbf{d}_i$        (8)
        $\mathbf{z}_{i+1} = \mathbf{H}\mathbf{g}_{i+1}$        (9)
        $\beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{z}_{i+1}}{\mathbf{g}_i^\top \mathbf{z}_i}$        (10)
        $\mathbf{d}_{i+1} = -\mathbf{z}_{i+1} + \beta_i \mathbf{d}_i$        (11)
    convergence check        if  $\mathbf{d}_{i+1}^\top \mathbf{d}_{i+1} <$ tolerance $\times\ \mathbf{d}_0^\top \mathbf{d}_0$ stop
end for
```

Special Case

If $\mathbf{H} = \mathbf{H}_m$, then in exact arithmetic (taking note that $\mathbf{H}_m \mathbf{H}_m = \mathbf{H}_m$)

$$\beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{z}_{i+1}}{\mathbf{g}_i^\top \mathbf{z}_i} = \frac{\mathbf{z}_{i+1}^\top \mathbf{z}_{i+1}}{\mathbf{z}_i^\top \mathbf{z}_i} \tag{12}$$

$$= \frac{\mathbf{z}_{i+1}^\top (\mathbf{z}_{i+1} - \mathbf{z}_i)}{\mathbf{z}_i^\top \mathbf{z}_i}. \tag{13}$$

Forms for the special cases of β_i in equations (12) and (13) are both found in [6], where $\mathbf{z}_i$ is evaluated implicitly via a quadratic linear constraints programming subproblem. We note that the only difference between [6] and the constrained steepest descent method [7] is in equation (11).

Other formulæ for β_i are also given in [6]. We note that they are not the same as the one given in this paper.

Since our formula for β_i is not found in the literature, it is prudent to prove that our algorithm produces mutually conjugate directions of search. We will also prove that the method converges in at most $n - m$ iterations in exact arithmetic.

THEOREM 1. *The directions of search determined by the proposed iteration are mutually conjugate.*

PROOF. The theorem can be stated in the form

$$\mathbf{d}_i^{\mathsf{T}} \mathbf{G} \mathbf{d}_j = 0, \qquad i \neq j, \quad i, j \geq 0.$$

The proof is by mathematical induction. We first assume that equation (11) produces mutually conjugate directions $\mathbf{d}_0, \dots, \mathbf{d}_k$. We shall prove that $\mathbf{d}_0, \dots, \mathbf{d}_{k+1}$ are mutually conjugate.

Using equations (8), (11), and the relation

$$\mathbf{d}_0 = -\mathbf{H}\mathbf{g}_0,$$

we get

$$\mathbf{d}_j = \sum_{i=0}^{j} a_i (\mathbf{HG})^i \mathbf{H}\mathbf{g}_0, \qquad 0 \leq j \leq k, \tag{14}$$

where $(\mathbf{HG})^0$ is defined to be $\mathbf{I}$.

First, we show that

$$\mathbf{g}_{k+1}^{\mathsf{T}} \mathbf{d}_j = 0. \tag{15}$$

For $j = k$,

$$\mathbf{g}_{k+1}^{\mathsf{T}} \mathbf{d}_k = 0,$$

since x_{k+1} minimizes f in the direction $\mathbf{d}_k$. For $0 \leq j \leq k - 1$,

$$\mathbf{g}_{k+1}^{\mathsf{T}} \mathbf{d}_j = \left(\mathbf{g}_{j+1}^{\mathsf{T}} + \sum_{i=j+1}^{k} \alpha_i \mathbf{d}_i^{\mathsf{T}} \mathbf{G} \right) \mathbf{d}_j = 0.$$

This completes the verification of (15).

In view of (15) and (14), $\mathbf{g}_{k+1}$ is also orthogonal to $\mathbf{H}\mathbf{g}_0, (\mathbf{HG})\mathbf{H}\mathbf{g}_0, \dots, (\mathbf{HG})^k \mathbf{H}\mathbf{g}_0$. Hence,

$$\mathbf{g}_{k+1}^{\mathsf{T}} (\mathbf{HG})^j \mathbf{H}\mathbf{g}_0 = 0, \qquad 0 \leq j \leq k. \tag{16}$$

For $1 \leq j \leq k$, we can write (16) in the form

$$(\mathbf{H}\mathbf{g}_{k+1})^{\mathsf{T}} \mathbf{G} (\mathbf{HG})^{j-1} \mathbf{H}\mathbf{g}_0 = 0.$$

Hence, $\mathbf{H}\mathbf{g}_{k+1}$ is mutually conjugate to $\mathbf{H}\mathbf{g}_0, (\mathbf{HG})\mathbf{H}\mathbf{g}_0, \dots, (\mathbf{HG})^{k-1}\mathbf{H}\mathbf{g}_0$.

In view of (14), $\mathbf{H}\mathbf{g}_{k+1}$ is also mutually conjugate to $\mathbf{d}_0, \dots, \mathbf{d}_{k-1}$. Setting $i = k$ in (11) and multiplying by $\mathbf{G}\mathbf{d}_j$, $j = 1, 2, \dots, k - 1$, we get

$$\begin{aligned} \mathbf{d}_{k+1}^{\mathsf{T}} \mathbf{G} \mathbf{d}_j &= -(\mathbf{H}\mathbf{g}_{k+1})^{\mathsf{T}} \mathbf{G}\mathbf{d}_j + \beta_k \mathbf{d}_k^{\mathsf{T}} \mathbf{G} \mathbf{d}_j \\ &= 0, \end{aligned}$$

and in the case $j = k$,

$$\begin{aligned} \mathbf{d}_{k+1}^{\mathsf{T}} \mathbf{G} \mathbf{d}_k &= -(\mathbf{H}\mathbf{g}_{k+1})^{\mathsf{T}} \mathbf{G}\mathbf{d}_k + \beta_k \mathbf{d}_k^{\mathsf{T}} \mathbf{G} \mathbf{d}_k \\ &= 0, \end{aligned} \tag{17}$$

since

$$\begin{aligned}\beta_k &= \frac{(\mathbf{H}\mathbf{g}_{k+1})^{\mathsf{T}}\mathbf{g}_{k+1}}{(\mathbf{H}\mathbf{g}_k)^{\mathsf{T}}\mathbf{g}_k} \\ &= \frac{\mathbf{g}_{k+1}^{\mathsf{T}}\mathbf{H}(\mathbf{g}_{k+1}-\mathbf{g}_k)}{-\mathbf{g}_k^{\mathsf{T}}\mathbf{H}(\mathbf{g}_{k+1}-\mathbf{g}_k)} \\ &= \frac{\mathbf{g}_{k+1}^{\mathsf{T}}\mathbf{H}\mathbf{G}\mathbf{d}_k}{-(\mathbf{H}\mathbf{g}_k)^{\mathsf{T}}\mathbf{G}\mathbf{d}_k} \\ &= \frac{(\mathbf{H}\mathbf{g}_{k+1})^{\mathsf{T}}\mathbf{G}\mathbf{d}_k}{(-\mathbf{H}\mathbf{g}_k+\beta_{k-1}\mathbf{d}_{k-1})^{\mathsf{T}}\mathbf{G}\mathbf{d}_k} \\ &= \frac{(\mathbf{H}\mathbf{g}_{k+1})^{\mathsf{T}}\mathbf{G}\mathbf{d}_k}{\mathbf{d}_k^{\mathsf{T}}\mathbf{G}\mathbf{d}_k}.\end{aligned}$$

We have shown that if our algorithm produces mutually conjugate directions $\mathbf{d}_0, \dots, \mathbf{d}_k$ then it produces mutually conjugate directions $\mathbf{d}_0, \dots, \mathbf{d}_{k+1}$.

It only remains to prove that $\mathbf{d}_0$ and $\mathbf{d}_1$ are conjugate directions and this is clear by putting $k = 0$ in (17).

The proof of the next theorem will use several properties of the method; the proofs of these properties are trivial, and hence, are not given.

Properties

(1) $\mathbf{C}^{\mathsf{T}}\mathbf{d}_k = \mathbf{0}$, this will ensure that every iteration point is feasible.
(2) $\mathbf{C}^{\mathsf{T}}\mathbf{H} = \mathbf{0}$.
(3) $\mathbf{d}_i$ are linearly independent.

THEOREM 2. *The proposed method minimizes f in (2) subject to the constraints given in (3) in at most $n - m$ iterations.*

PROOF. Let $s = n - m$. Applying (7) repeatedly, we have

$$\mathbf{x}_s = \mathbf{x}_{r+1} + \sum_{q=r+1}^{s-1} \alpha_q \mathbf{d}_q, \qquad 0 \le r < s-1.$$

The gradient of f takes the form

$$\mathbf{g}_s = \mathbf{g}_{r+1} + \sum_{q=r+1}^{s-1} \alpha_q \mathbf{G}\mathbf{d}_q,$$

$$\mathbf{g}_s^{\mathsf{T}}\mathbf{d}_r = \mathbf{g}_{r+1}^{\mathsf{T}}\mathbf{d}_r + \sum_{q=r+1}^{s-1} \alpha_q \mathbf{d}_q^{\mathsf{T}}\mathbf{G}\mathbf{d}_r.$$

Using Theorem 1 and the fact that $\mathbf{g}_{r+1}^{\mathsf{T}}\mathbf{d}_r = 0$, we get

$$\mathbf{g}_s^{\mathsf{T}}\mathbf{d}_r = 0, \tag{18}$$

and

$$\mathbf{g}_s^{\mathsf{T}}\mathbf{d}_{s-1} = 0,$$

since α_{s-1} is determined to minimize f along the direction $\mathbf{d}_{s-1}$.

Let the linear operator $\mathbf{T} : \mathbb{R}^n \to \mathbb{R}^m$ be multiplication by $\mathbf{C}^{\mathsf{T}}$, where $\mathbb{R}^n$ and $\mathbb{R}^m$ are n-dimensional and m-dimensional vector spaces. Properties 1 and 2 show that

$$\mathbf{d}_k, \mathbf{H}\mathbf{d}_k \in \ker \mathbf{T},$$

where $\ker \mathbf{T}$ is the kernel of $\mathbf{T}$.

Hence, we can write

$$\mathbf{Hd}_k = \sum_{i=0}^{s-1} \lambda_i \mathbf{d}_i, \qquad 0 \le k \le s-1, \tag{19}$$

since $\mathbf{d}_i$ are linearly independent.

By virtue of (18) and (19), and using the symmetry of $\mathbf{H}$, we have

$$(\mathbf{Hg}_s)^{\mathrm{T}} \mathbf{d}_k = 0. \tag{20}$$

Equation (20) implies that

$$\mathbf{Hg}_s = 0,$$

since $\mathbf{Hg}_s \in \ker \mathbf{T}$.

The minimum may be obtained earlier if λ_i all turn out to be zero at the end part of the iterations.

We shall now derive a rate of convergence for the constrained conjugate gradient method.

THEOREM 3. *For the constrained conjugate method, we have the error estimate*[1]

$$\|\mathbf{x}_k - \mathbf{x}^*\|_G \le T_k \left(\frac{\lambda_{n-m} + \lambda_1}{\lambda_{n-m} - \lambda_1} \right)^{-1} \|\mathbf{x}_0 - \mathbf{x}^*\|_G,$$

where

$$\|\mathbf{x}\|_G = \left(\mathbf{x}^{\mathrm{T}} \mathbf{Gx}\right)^{1/2},$$

T_k *is the Chebyshev polynomial of degree* k *and* λ_{n-m} *and* λ_1 *are the largest and smallest eigenvalues of* $\mathbf{Z}^{\mathrm{T}}\mathbf{GZ}$*, respectively.*

Further, if $\rho(\varepsilon)$ *is defined for any* $\varepsilon > 0$ *to be the smallest integer* k *such that*

$$\|\mathbf{x}_k - \mathbf{x}^*\|_G \le \varepsilon \|\mathbf{x}_0 - \mathbf{x}^*\|_G,$$

then

$$\rho(\varepsilon) \le \frac{1}{2}\sqrt{C(\mathbf{Z}^{\mathrm{T}}\mathbf{GZ})} \ln \frac{2}{\varepsilon} + 1,$$

where $C(\mathbf{Z}^{\mathrm{T}}\mathbf{GZ})$ *is the condition number of* $\mathbf{Z}^{\mathrm{T}}\mathbf{GZ}$.

PROOF. Consider a transformation of variable by the relation

$$\mathbf{x} = \mathbf{Zy} + \mathbf{Wq}^*. \tag{21}$$

Hence, (21) can be considered as elimination of constraints by transformation of variable and it is obvious that any $\mathbf{x}$ given by (21) is feasible.

Substituting (21) in (2), we get

$$f = \frac{\mathbf{y}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{GZy}}{2} + \mathbf{y}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}(\mathbf{GWq}^* - \mathbf{w}) + \text{ constant}. \tag{22}$$

The minimization of (2) subject to constraint in (3) is equivalent to the unconstrained minimization of (22) for $n - m$ variables.

If we apply the Fletcher and Reeves [5] algorithm in (22), we get

```
initialise
```

$$\mathbf{y} = \mathbf{y}_0$$

$$\hat{\mathbf{g}}_0 (\equiv \mathbf{Z}^{\mathrm{T}}\mathbf{g}_0) = \mathbf{Z}^{\mathrm{T}}\mathbf{GZy}_0 + \mathbf{Z}^{\mathrm{T}}(\mathbf{GWq}^* - \mathbf{w}) \tag{23}$$

$$\hat{\mathbf{d}}_0 (\equiv -\mathbf{Z}^{\mathrm{T}}\mathbf{g}_0) = -\hat{\mathbf{g}}_0 \tag{24}$$

---

[1] An asterisk will always denote an optimal value.

```
for $i = 0, 1, 2$
    $\alpha_i = \frac{-\hat{\mathbf{g}}_i^T \hat{\mathbf{d}}_i}{\hat{\mathbf{d}}_i^T \mathbf{Z}^T \mathbf{G} \mathbf{Z} \hat{\mathbf{d}}_i} \left( \equiv \frac{-\mathbf{g}_i^T \mathbf{d}_i}{\mathbf{d}_i^T \mathbf{G} d_i} \right),$
    (where $\mathbf{d}_i = \mathbf{Z}\mathbf{d}_i$and $\hat{\mathbf{g}}_i = \mathbf{Z}^T \mathbf{g}_i$),
    $\mathbf{y}_{i+1} = \mathbf{y}_i + \alpha_i \hat{\mathbf{d}}_i$    (25)
    $\hat{\mathbf{g}}_{i+1} = \mathbf{Z}^T \mathbf{g}_{i+1}$    (26)
    $\beta_i = \frac{\hat{\mathbf{g}}_{i+1}^T \hat{\mathbf{g}}_{i+1}}{\hat{\mathbf{g}}_i \hat{\mathbf{g}}_i} \left( \equiv \frac{\mathbf{g}_{i+1}^T \mathbf{z}_{i+1}}{\mathbf{g}_i^T \mathbf{z}_i} \right)$
    $\hat{\mathbf{d}}_{i+1} = -\hat{\mathbf{g}}_{i+1} + \beta_i \hat{\mathbf{d}}_i$    (27)
    if $\hat{\mathbf{d}}_{i+1}^T \hat{\mathbf{d}}_{i+1} <$ tolerance $\times\ \hat{\mathbf{d}}_0^T \hat{\mathbf{d}}_0$ stop
end for
```

The above algorithm can be immediately transformed to our algorithm by multiplying equations (23)–(27) by $\mathbf{Z}$ and taking

$$\mathbf{x}_0 = \mathbf{Z}\mathbf{y}_0 + \mathbf{W}\mathbf{q}^*.$$

We note that the $\mathbf{y}_k$ values are related to the $\mathbf{x}_k$ values of our algorithm via equation (21).

An error estimate for the above algorithm is well known [8], i.e.,

$$\|\mathbf{y}_k^* - \mathbf{y}^*\|_{\mathbf{Z}^T\mathbf{G}\mathbf{Z}} \le T_k \left(\frac{\lambda_{n-m} + \lambda_1}{\lambda_{n-m} - \lambda_1} \right)^{-1} \|\mathbf{y}_0 - \mathbf{y}^*\|_{\mathbf{Z}^T\mathbf{G}\mathbf{Z}}, \tag{28}$$

where λ_{n-m} and λ_1 are the largest and smallest eigenvalues of $\mathbf{Z}^T\mathbf{G}\mathbf{Z}$, respectively, and if $\rho(\varepsilon)$ is defined for any $\varepsilon > 0$ to be the smallest integer k such that

$$\|\mathbf{y}_k - \mathbf{y}^*\|_{\mathbf{Z}^T\mathbf{G}\mathbf{Z}} \le \varepsilon \|\mathbf{y}_0 - \mathbf{y}^*\|_{\mathbf{Z}^T\mathbf{G}\mathbf{Z}}, \tag{29}$$

then

$$\rho(\varepsilon) \le \frac{1}{2} \sqrt{C(\mathbf{Z}^T\mathbf{G}\mathbf{Z})} \ln \frac{2}{\varepsilon} + 1.$$

It can be easily shown that

$$\|\mathbf{x}_k - \mathbf{x}^*\|_{\mathbf{G}} = \|\mathbf{y}_k - \mathbf{y}^*\|_{\mathbf{Z}^T\mathbf{G}\mathbf{Z}}. \tag{30}$$

Substitution of (30) in (28) and (29) completes the proof.

3. SOLVING AX = B VIA THE PROPOSED METHOD

The state of the art for solving (1) is not nearly as advanced as that for solving a positive definite system and there is a good deal of current research being carried out on this topic. We do not intend, in the main, to discuss other algorithms and compare them with the proposed algorithm. However, in Section 4, an existing algorithm is compared with the proposed algorithm.

We tackle problem (1) indirectly by considering

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{31}$$

as a set of constraints (see below). It is obvious that if $\mathbf{x}^*$ satisfies the constraint (31), then $\mathbf{x}^*$ is the solution of (1). The solution of (1) is obtained by minimising a quadratic function

$$f = \frac{\mathbf{s}^T\mathbf{s}}{2} \tag{32a}$$

subject to the constraint

$$\mathbf{Ax} + \varepsilon\mathbf{s} = \mathbf{b}, \tag{32b}$$

where ε is a real parameter and $\mathbf{s}$ is an $n \times 1$ vector variables. For simplicity we use the orthogonal projection matrix in our algorithm, i.e., $\mathbf{ZZ}^{\mathsf{T}} = \mathbf{H}_m$. The matrix $\mathbf{H}_m$ is obtained similar to that given in [3], i.e.,

$$\mathbf{H}_e = \mathbf{H}_{e-1} - \frac{\mathbf{H}_{e-1}\mathbf{c}_e\mathbf{c}_e^{\mathsf{T}}\mathbf{H}_{e-1}}{\mathbf{c}_e^{\mathsf{T}}\mathbf{H}_{e-1}\mathbf{c}_e}, \qquad e = 1, 2, \dots, m = n, \tag{33}$$

where $\mathbf{H}_0 = \mathbf{I}$ (the $2n \times 2n$ identity matrix), $\mathbf{c}_e$ are the columns of $\mathbf{C} = \left(\mathbf{A}^{\mathsf{T}}/_{\varepsilon\mathbf{I_n}}\right)$ and $\mathbf{I_n}$ is the $n \times n$ identity matrix.

Let

$$\mathbf{t} = \begin{pmatrix} \mathbf{x} \\ \mathbf{s} \end{pmatrix}$$

be the variable for the problem (32). The solution of (32) is obviously

$$\mathbf{t}^* = \begin{pmatrix} \mathbf{x}^* \\ \mathbf{0} \end{pmatrix},$$

and since $\mathbf{x}^*$ satisfies (31), it must be the solution of (1). Numerical solution of problem (32) is obtained via the proposed conjugate gradient method developed above.

At the onset, one might say what is the difference between solving problem (1) via our proposed method and solving[2]

$$\mathbf{A}^{\mathsf{T}}\mathbf{Ax} = \mathbf{A}^{\mathsf{T}}\mathbf{b} \tag{34}$$

via the unconstrained Fletcher and Reeves [5] conjugate gradient method which has already been established.

(Note that equation (34) can be simply be obtained from minimising

$$f = \frac{(\mathbf{b} - \mathbf{Ax})^{\mathsf{T}}(\mathbf{b} - \mathbf{Ax})}{2}, \tag{35}$$

where f in (35) is obtained by letting $\varepsilon = 1$ in (32b), write

$$\mathbf{s} = \mathbf{b} - \mathbf{Ax}, \tag{36}$$

and substitute (36) in (32a)).

The difference is fundamental in the sense that if the same starting point $\mathbf{x}_0$ is used, the sequence of points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots$ generated by the proposed method are different from the unconstrained version which is based on minimising (35). We note that our $\mathbf{x}$-component directions of search are also different from the directions of search produced by the unconstrained algorithm.

Note that in our computer program our initial values of $\mathbf{t}$ and $\mathbf{g}$ are

$$\mathbf{t}_0 = \begin{pmatrix} \mathbf{0} \\ \mathbf{b}/\varepsilon \end{pmatrix},$$

and

$$\mathbf{g}_0 = \begin{pmatrix} \mathbf{0} \\ \mathbf{s}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b}/\varepsilon \end{pmatrix},$$

respectively. The term α_i in equation (6) takes a simpler form, i.e.,

$$\alpha_i = \frac{-\mathbf{g}_i^{\mathsf{T}}\mathbf{d}_i}{\tilde{\mathbf{d}}_i^{\mathsf{T}}\tilde{\mathbf{d}}_i},$$

---

[2] We note that $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ has a condition number equal to the square of the condition number of $\mathbf{A}$.

where the n-vector $\tilde{\mathbf{d}}_i$ contains the last n elements of the $2n$-vector $\mathbf{d}_i$. The scalar ε plays an important role in convergence. In the s-space, the contours of f are circular. Geometrically, as $\varepsilon \to 0$ the flat $\mathbb{F}_m$ tends to position itself 'parallel' to the s-space, and the contours of f in the flat $\mathbb{F}_m$ tend to be circular; this leads to faster convergence.

Below is an analytical proof that as $\varepsilon \to 0$, the solution of (32) can be obtained in exactly one iteration.

PROOF. $\mathbf{H}_m$ can be written as (see, for example, [9])

$$\mathbf{H}_m = \mathbf{I} - \mathbf{C}\left(\mathbf{C}^{\mathsf{T}}\mathbf{C}\right)^{-1}\mathbf{C}^{\mathsf{T}}.$$

Here, $\mathbf{C}^{\mathsf{T}} = (\mathbf{A}|\varepsilon\mathbf{I_n})$. Retaining terms up to $O(\varepsilon^2)$, we have

$$\mathbf{H}_m = \left(\begin{array}{c|c} \varepsilon^2\mathbf{N} & -\varepsilon\mathbf{A}^{-1} \\ \hline -\varepsilon(\mathbf{A}^{-1})^{\mathsf{T}} & \mathbf{I_n} + \varepsilon^2\mathbf{R} \end{array}\right),$$

where $\mathbf{N} = (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}$ and $\mathbf{R} = -(\mathbf{A}\mathbf{A}^{\mathsf{T}})^{-1}$.

The initial direction takes the form

$$\mathbf{d}_0 = \begin{pmatrix} \mathbf{A}^{-1}\mathbf{b} \\ -\mathbf{b}/\varepsilon + \varepsilon\mathbf{R}\mathbf{b} \end{pmatrix}.$$

Consider the first iteration

$$\begin{aligned} \mathbf{t}_1 &= \mathbf{t}_0 + \alpha_0\mathbf{d}_0 \\ &= \begin{pmatrix} \mathbf{A}^{-1}\mathbf{b} + \varepsilon^2\xi\mathbf{A}^{-1}\mathbf{b} \\ -\varepsilon\left[\mathbf{R}\mathbf{b} + \xi\mathbf{b}\right] + \mathrm{O}\left(\varepsilon^3\right) \end{pmatrix} \end{aligned}$$

where $\alpha_0 = 1 + \varepsilon^2\xi$ and $\xi = -\mathbf{b}^{\mathsf{T}}\left(\mathbf{A}\mathbf{A}^{\mathsf{T}}\right)^{-1}\mathbf{b} - \mathbf{b}^{\mathsf{T}}\mathbf{R}\mathbf{b}$.

Hence, as $\varepsilon \to 0$

$$\mathbf{t}_1 = \mathbf{t}^* = \begin{pmatrix} \mathbf{x}^* \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{A}^{-1}\mathbf{b} \\ \mathbf{0} \end{pmatrix}.$$

This completes the proof.

However, in practice, using a large value of $1/\varepsilon$ is sufficient to obtain numerical results in one iteration. See Section 4 for more details.

Next, we show that the condition number $C(\mathbf{Z}^{\mathsf{T}}\mathbf{G}\mathbf{Z}) = 1$ for $\mathbf{H}_m$ approximated up to $\mathrm{O}(\varepsilon)$, i.e.,

$$\mathbf{H}_m = \left(\begin{array}{c|c} \mathbf{0} & -\varepsilon\mathbf{A}^{-1} \\ \hline -(\varepsilon\mathbf{A}^{-1})^{\mathsf{T}} & \mathbf{I_n} \end{array}\right). \tag{37}$$

Let $\mathbf{Z} = \left(-\varepsilon\mathbf{A}^{-1}/\mathbf{I_n}\right)$ (taking note that $(\mathbf{A}|\varepsilon\mathbf{I_n})\mathbf{Z} = \mathbf{0}$). Up to $O(\varepsilon)$, $\mathbf{Z}\mathbf{Z}^{\mathsf{T}} = \mathbf{H}_m$. Since $\mathbf{G} = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I_n} \end{array}\right)$, we have $\mathbf{Z}^{\mathsf{T}}\mathbf{G}\mathbf{Z} = \mathbf{I_n}$. Hence, $C(\mathbf{Z}^{\mathsf{T}}\mathbf{G}\mathbf{Z}) = C(\mathbf{I_n}) = 1$.

Obtaining the solution in one iteration suggests that we can obtain $\mathbf{A}^{-1}$ in a finite number of steps using equation (33) alone, noting that up to $\mathrm{O}(\varepsilon)$, $\mathbf{H}_m$ takes the form given in (37), $\mathbf{A}^{-1}$ is contained in $\mathbf{H}_m$ and $\mathbf{H}_e$ takes the form

$$\mathbf{H}_e = \left(\begin{array}{c|c} \mathbf{J}_e & \varepsilon\mathbf{K}_e \\ \hline \varepsilon\mathbf{K}_e^{\mathsf{T}} & \mathbf{I_n} \end{array}\right), \qquad e = 0, 1, \ldots, n = m,$$

with $\mathbf{J}_0 = \mathbf{I_n}$, $\mathbf{K}_0 = \mathbf{0}$. From (37), $\mathbf{K}_n = -\mathbf{A}^{-1}$ and $\mathbf{J}_n = \mathbf{0}$. We note that the last $n - e$ columns of $\mathbf{K}_e$ have zero elements and $\mathbf{J}_e$ and $\mathbf{K}_e$ are independent of ε. We are now in a position to state a direct method (finite number of steps) of finding $\mathbf{A}^{-1}$. The method is[3]

---

[3] This method may be unstable for certain matrix types.

```
initialise
    $\mathbf{J}_0 = \mathbf{I}_n$
    $\mathbf{K}_0 = \mathbf{0}$
for $i = 1, 2, \ldots, n$; compute
    ${}^J\mathbf{t}_i = \mathbf{J}_{i-1}\mathbf{a}_i$
    where the $n$-vector $\mathbf{a}_i$ is the $i$th column of $\mathbf{A}^\top$
    $\gamma_i = \mathbf{a}_i^\top {}^J\mathbf{t}_i$
    ${}^J\mathbf{s}_i = {}^J\mathbf{t}_i / \gamma_i$                                   (38)
    $\mathbf{J}_i = \mathbf{J}_{i-1} - {}^J\mathbf{t}_i {}^J\mathbf{s}_i^\top$
    ${}^K\mathbf{t}_i = (\mathbf{K}_{i-1})^\top \mathbf{a}_i + \mathbf{e}_i$
    where $\mathbf{e}_i$ is the standard basis of $\mathbb{R}^n$
    $\mathbf{K}_i = \mathbf{K}_{i-1} - {}^J\mathbf{s}_i {}^K\mathbf{t}_i^\top$
end for
```

It can be easily shown that $\gamma_i > 0$ if $\mathbf{A}$ is nonsingular. The above method involves matrix-vector multiplications and outer products which are easily parallelizable. This has an advantage over existing well-known direct methods. Discussion on the difficulties in parallelizing existing direct methods is found in [10]. For example, it is quoted in [10] that the highest reported speed-up for some well-known direct methods did not exceed a value of about 3 or 4 even when many processors were used on large MIMD parallel networks.

The number of multiplications (here the word multiplication is used for either multiplication or division) needed to compute $\mathbf{A}^{-1}$ using Gaussian elimination without pivoting is n^3 which is better than our algorithm which is $(5n^3/2)$ + lower degree terms (after taking account of symmetry and zero elements). However, pivoting is not needed in our method, and for sparse matrices considerable savings can be obtained. Research on this is currently being done (see [11], for example).

4. NUMERICAL RESULTS

Comparisons of various iterative methods have been carried out recently [12] . In the past, many opinions have been expressed on the issue of which class of methods is better; however, iterative methods (and also direct methods) have been developing so rapidly in the recent past that it is unlikely that definite comparisons are possible even now. An existing iterative method is chosen to give a rough comparison of its performance against ours. We note that the problem sample is quite small, so the results might be misleading; a more rigorous numerical experiment is needed and it is an interesting area of further research. As mentioned earlier, in this paper we only provide an alternative method of solving (1); the method was primarily developed for solving linear constraint finite element problems (see [11], for example). We make no claim that it is better than other existing iterative methods (or direct methods).

The iterative method included to compare with ours is the standard conjugate acceleration (Bi-conjugate) method adapted to nonsymmetric matrices [12]. We have no intention to compare with direct methods. All programs are written in Microsoft® Fortran 77 and are double precision. Solutions are obtained when $\|\mathbf{d}_{i+1}\| < 10^{-6}\|\mathbf{d}_0\|$. Computations are carried out on a Viglen 286 personal computer without coprocessor.

In the experiment, two arbitrary matrices, a 20 × 20 and 40 × 40, are used. They appear on page 35 and they are not diagonally dominant. The elements of $\mathbf{b}$ have unit values.

$$\begin{pmatrix}
0 & 1 & -1 & 1 & 0 & 3 & 4 & 0 & 0 & 1 & -1 & -2 & 0 & 3 & 4 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & -2 & -4 & 0 & 2 & 8 & 3 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -2 & 3 & 0 & 0 & 3 & 1 & -1 & 4 & 6 & 6 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 2 & 2 & 1 & 2 & 1 & 0 & -1 & -1 & 3 & 2 & -2 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 2 & -2 & -2 & -2 & 0 & 1 & 4 & 6 & 10 & 0 & 0 & 0 & 0 & 1 \\
-1 & -1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & 0 & 0 & 1 & 0 & 0 \\
-1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 1 & 3 & -1 & 2 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & -1 & 2 & -1 & 1 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 1 & 2 & 3 & 4 & 5 & 6 & -1 & 0 & 0 & 0 & 0 \\
9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & -1 & 0 & 0 & 0 \\
-1 & 2 & -3 & 4 & -5 & 6 & -7 & 8 & -9 & 0 & 0 & 9 & -8 & 7 & -5 & 0 & 0 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 \\
-1 & -1 & -1 & -2 & -2 & -2 & -3 & -3 & -3 & 0 & -4 & -4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & -1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 2 & -1 & 1 & -3 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 8 & 8 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 1 & 3 & -3 & 3 & -3 & 4 & 4 & 4 & 5 & 5 & 7 & 0 & 0 & 1 & 0 & 0 \\
9 & 0 & -9 & 0 & 0 & 0 & -1 & 2 & 2 & 0 & -5 & 2 & -7 & 6 & 0 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

and

$$\begin{pmatrix}
5 & 3 & -3 & 1 & 0 & 3 & 4 & 0 & 0 & 3 & -1 & -2 & 0 & 3 & 4 & 3 & 5 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 3 & 4 & 0 & 0 & 1 & -1 & -2 & 0 & 3 & 4 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & -2 & -4 & 0 & 2 & 8 & 3 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -2 & 3 & 0 & 0 & 3 & 1 & -1 & 4 & 6 & 6 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 2 & 2 & 1 & 2 & 1 & 0 & -1 & -1 & 3 & 2 & -2 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 2 & -2 & -2 & -2 & 0 & 1 & 4 & 6 & 10 & 0 & 0 & 0 & 0 & 1 \\
-1 & -1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & 0 & 0 & 1 & 0 & 0 \\
-1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 1 & 2 & 3 & 4 & 5 & 6 & -1 & 0 & 0 & 0 & 0 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & -1 & 0 & 0 & 0 \\
-1 & 2 & -3 & 4 & -5 & 6 & -7 & 8 & -9 & 0 & 0 & 9 & -8 & 7 & -5 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 4 \\
-1 & -1 & -1 & -2 & -2 & -2 & -3 & -3 & -3 & 0 & -4 & -4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & -1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 8 & 8 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 3 & -3 & 3 & -3 & 4 & 4 & 4 & 5 & 5 & 7 & 0 & 0 & 1 & 0 & 0 \\
9 & 0 & -9 & 0 & 0 & 0 & -1 & 2 & 2 & 0 & -5 & 2 & -7 & 6 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 3 & 1 & -2 & -4 & 5 & 2 & 8 & 3 & -1 & 0 & 0 & 3 & 0 & 0 & 0 \\
0 & 1 & 5 & 0 & 1 & -2 & 3 & 0 & 0 & 3 & 3 & -1 & 4 & 6 & 6 & 0 & 0 & 1 & 0 & 5 & 0 & 0 & 3 & 0 & 2 & 2 & 1 & 2 & 1 & 0 & -3 & -1 & 3 & 2 & -2 & 0 & 0 & 5 & 3 & 0 \\
1 & 1 & 3 & 0 & 0 & 0 & 2 & -2 & -2 & -2 & 0 & 1 & 4 & 6 & 35 & 0 & 0 & 0 & 0 & 1 & -3 & -3 & 1 & 5 & 0 & 3 & 0 & 1 & 3 & 3 & 3 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & 0 & 5 & 3 & 0 & 0 & -1 & 0 & -3 & 0 & -1 & 0 & -1 & 0 & -3 & 0 & -1 & 0 & -3 & 0 & -1 & 0 & 3 & 0 & 5 & 0 \\
0 & 1 & 0 & 1 & 5 & 3 & 0 & 1 & 0 & 1 & 0 & 1 & 5 & 1 & 0 & 1 & 0 & 0 & 5 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 1 & 2 & 3 & 4 & 5 & 6 & -3 & 0 & 0 & 0 & 0 \\
9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 5 & 3 & 2 & 3 & 4 & 5 & 0 & -1 & 0 & 5 & 0 & -1 & 2 & -3 & 4 & -5 & 6 & -7 & 8 & -9 & 0 & 0 & 9 & -8 & 7 & -5 & 0 & 5 & -3 & 0 & 0 \\
1 & 5 & 0 & 0 & 0 & 5 & 0 & 0 & 5 & 0 & 0 & 5 & 0 & 0 & 1 & 5 & 0 & 0 & -3 & 0 & -1 & -3 & -1 & -2 & -2 & -2 & -3 & -3 & -3 & 0 & -4 & -4 & -4 & 0 & 0 & 5 & 0 & 5 & 0 & -1 \\
0 & 5 & 0 & 5 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & -3 & -1 & 3 & -1 & 3 & -1 & 1 & -1 & 3 & -1 & 3 & -1 & 1 & -3 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
0 & 0 & 2 & 0 & 5 & 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3 & -1 & 0 & 0 & 5 & 0 & 5 & 0 & 0 & 8 & 8 & 8 & 8 & 0 & 0 & 5 & 0 & 0 & 5 & 5 & 0 \\
0 & -1 & 0 & 3 & 1 & 3 & -3 & 3 & -3 & 4 & 4 & 4 & 5 & 5 & 7 & 0 & 0 & 1 & 5 & 0 & 9 & 0 & -9 & 0 & 0 & 0 & -1 & 2 & 2 & 0 & - & 52 & -7 & 6 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & -1 & 1 & 4 & 0 & 4 & 0 & 2 & 1 & -1 & -2 & 0 & 0 & 4 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & -2 & -4 & 0 & 2 & 8 & 0 & -1 & 2 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 2 & 0 & 1 & -2 & 0 & 0 & 0 & 2 & 1 & -1 & 4 & 6 & 6 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 2 & 2 & 1 & 2 & 1 & 2 & -1 & -1 & 0 & 2 & -2 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 2 & -2 & -2 & -2 & 0 & 1 & 4 & 6 & 10 & 0 & 0 & 2 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 2 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 2 & 0 & 0 & 1 & 2 & 0 & 4 & 5 & 6 & 7 & 8 & 9 & 1 & 2 & 0 & 4 & 5 & 6 & -1 & 0 & 2 & 0 & 0 \\
9 & 8 & 7 & 6 & 5 & 4 & 0 & 2 & 1 & 0 & 1 & 2 & 0 & 4 & 5 & 0 & -1 & 0 & 2 & 0 & -1 & 2 & -0 & 4 & -5 & 6 & -7 & 8 & -9 & 0 & 0 & 9 & -8 & 7 & -5 & 0 & 0 & -1 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -2 & -2 & -2 & -0 & -0 & -0 & 0 & -4 & -4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
0 & 0 & 2 & 0 & 2 & 2 & 0 & -4 & 2 & 0 & 0 & 1 & -4 & 0 & 1 & 0 & 0 & 0 & 1 & -0 & -1 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 8 & 8 & 8 & 8 & -4 & 0 & 0 & 0 & -4 & 0 & 0 & -4 \\
0 & -1 & 0 & 1 & 1 & 0 & -0 & 0 & -0 & 4 & 4 & 4 & 5 & 5 & 7 & 0 & 0 & 1 & -4 & 0 & 9 & 0 & -9 & 0 & 0 & 0 & -1 & 2 & 2 & 0 & -5 & 2 & -7 & 6 & 0 & 1 & 1 & 1 & 1 & 1 \\
5 & 6 & -1 & 1 & 4 & 0 & 4 & 0 & 2 & 1 & -6 & -2 & 0 & 0 & 4 & 1 & 0 & 2 & 5 & 0 & 1 & 0 & 6 & 1 & 1 & 1 & 6 & -2 & -4 & 5 & 2 & 1 & 0 & -6 & 2 & 0 & 1 & 0 & 0 & 5 \\
0 & 6 & 2 & 0 & 1 & -2 & 5 & 0 & 0 & 2 & 1 & -6 & 4 & 6 & 6 & 5 & 0 & 1 & 2 & 5 & 0 & 0 & 6 & 0 & 2 & 2 & 1 & 2 & 6 & 2 & -1 & -6 & 5 & 2 & -2 & 0 & 0 & 5 & 1 & 0 \\
1 & 6 & 1 & 5 & 0 & 0 & 2 & -2 & -2 & -2 & 5 & 6 & 4 & 6 & 10 & 5 & 0 & 2 & 5 & 1 & -6 & -1 & 6 & 0 & 5 & 1 & 0 & 6 & 1 & 1 & 6 & 1 & 6 & 1 & 6 & 0 & 5 & 0 & 1 & 0 \\
1 & 6 & 1 & 6 & 1 & 6 & 0 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & 0 & 5 & 1 & 0 & 5 & -1 & 0 & -1 & 0 & -1 & 5 & -1 & 0 & -6 & 0 & -1 & 0 & -1 & 0 & -6 & 5 & 1 & 0 & 2 & 0 \\
5 & 1 & 0 & 6 & 5 & 1 & 0 & 1 & 5 & 6 & 0 & 1 & 5 & 1 & 0 & 1 & 5 & 2 & 0 & 0 & 6 & 2 & 0 & 4 & 5 & 6 & 7 & 1 & 9 & 6 & 2 & 0 & 4 & 5 & 6 & -1 & 5 & 2 & 0 & 0 \\
9 & 1 & 7 & 6 & 5 & 4 & 5 & 2 & 6 & 0 & 1 & 2 & 5 & 4 & 5 & 0 & -6 & 5 & 2 & 0 & -1 & 2 & -0 & 4 & -5 & 6 & -7 & 1 & -9 & 5 & 0 & 9 & -1 & 7 & -5 & 0 & 0 & -1 & 0 & 0 \\
1 & 0 & 2 & 5 & 0 & 2 & 0 & 0 & 2 & 5 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -6 & -2 & -2 & -2 & -0 & -5 & -0 & 0 & -4 & -4 & -4 & 5 & 0 & 5 & 0 & 0 & 0 & -1 \\
0 & 5 & 0 & 0 & 1 & 6 & 5 & 0 & 5 & 0 & 0 & 5 & 0 & 1 & 0 & -1 & 0 & -1 & 5 & -1 & -1 & 6 & -1 & 1 & -1 & 1 & -6 & 1 & -1 & 6 & -1 & 1 & -6 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
0 & 5 & 2 & 0 & 2 & 2 & 0 & -4 & 2 & 0 & 5 & 1 & -4 & 0 & 1 & 5 & 0 & 5 & 6 & -0 & -1 & 0 & 0 & -4 & 5 & 0 & 5 & 0 & 1 & 1 & 6 & 8 & -4 & 0 & 5 & 0 & -4 & 0 & 0 & -4 \\
5 & -7 & 0 & 7 & 7 & 5 & -0 & 5 & -0 & 4 & 4 & 4 & 5 & 5 & 7 & 0 & 5 & 1 & -4 & 0 & 9 & 0 & -9 & 0 & 0 & 0 & -1 & 2 & 2 & 0 & -5 & 2 & -7 & 6 & 0 & 1 & 7 & 7 & 1 & 1
\end{pmatrix}$$

Table 1. Number of iterations and CPU time.

No. of equations	Solver	Constrained conjugate gradient $\varepsilon = 1.0E - 20/\varepsilon = 1.0E - 10$	Bi-conjugate
20	Number of	1	23
40	iterations	1	80
20	CPU time	13	16
40	seconds	95	178

The results are tabulated in Table 1. The total required storage, excluding **A**, **x**, and **b** are not given because efficient storage programs are not written. We feel that, due to this inefficiency, the reader may have the wrong impression if the storage values are given.

Table 1 indicates that our method seems better for the above matrices.

In our method, the solutions are obtained in one iteration when $\varepsilon = 1.0E - 20$ and $1.0E - 10$. This supports the theory given in Section 3. For the smaller matrix, iteration counts for $\varepsilon = 1$ and $10^{-1} \leq \varepsilon \leq 10^{-20}$ are 7 and 1, respectively, and for the larger matrix, numbers of iterations for $\varepsilon = 1, 10^{-2}, 10^{-3} \leq \varepsilon \leq 10^{-20}$ are 3177, 2 and 1, respectively. We note that for very large n and when a solution is obtained in one iteration, for some small suitable values of ε, the number of multiplications is about $(5n^3/2)$. Hence, for large n, although a solution can be obtained in one iteration, the method can be expensive, unless **A** is sparse or the calculations are worked out in parallel.

Results for varying the right-hand side are tabulated in Table 2. The elements of the second right-hand side and third right-hand side vectors take the values of two and three, respectively.

Table 2. Number of iterations and CPU time for varying right-hand sides after the solution for the first right-hand side is obtained. The table is for both $b_i = 2$ and $b_i = 3$, where b_i are the elements of **b**.

No. of equations	Solver	Constrained conjugate gradient $\varepsilon = 1.0E - 20/\varepsilon = 1.0E - 10$	Bi-conjugate
20	Number of	1	23
40	iterations	1	80
20	CPU time	1	16
40	seconds	3	178

From Table 2, we can safely conclude that our method is better in the case when the right-hand side is varied.

When tested on the ill-conditioned Hilbert matrices (for $n = 3$ to 11), with $\varepsilon = 1.0E - 20$, solutions are obtained in one iteration. However, the accuracy of solutions, within four significant figures, deteriorates as n increases. For example, for $n = 11$, although the solution is obtained in one iteration using $\varepsilon = 1.0E - 20$, the answer is inaccurate. This is due to rounding errors in calculating the $\mathbf{H}_m$ matrix which produced directions of search in a flat $\tilde{\mathbb{F}}_m$ different from the exact flat $\mathbb{F}_m$. Note that for nearly singular matrices a slight change in one of the hyperplanes, i.e., one of the constraints, gives a large change in the solution.

The solutions can also be obtained using the direct method given in (38). The CPU timings are similar to those of the iterative method of $\varepsilon = 1.0E - 20$ and $\varepsilon = 1.0E - 10$.

Finally, we note that more experiments and research are needed to be done to investigate the solving of equation (1) via our methods. For example, research is needed to investigate the method when the direction of search is obtained implicitly or to investigate the use of different forms of **Z**, to realize their potential for large-scale problems.

REFERENCES

1. R. Fletcher and M.J.D. Powell, A rapidly convergent descent method for minimization, *The Computer Journal* **6**, 163–168 (1963).
2. D. Goldfarb and L. Lapidus, Conjugate gradient method for nonlinear programming problems with linear constraints, *Indust. and Eng. Chem. Fundamentals* **7**, 142–157 (1968).
3. D. Goldfarb, Extension of Davidon's variable metric method to maximization under linear inequality and equality constraints, *SIAM Journal on Applied Mathematics* **17**, 739–764 (1969).
4. R. Fletcher, A new approach to variable metric algorithms, *The Computer Journal* **13**, 317–322 (1970).
5. R. Fletcher and C.M. Reeves, Function minimization by conjugate gradients, *The Computer Journal* **7**, 149–154 (1964).
6. J.S. Arora and G. Li, Constrained conjugate directions methods for design optimization of large systems, *A.I.A.A.J.* **31**, 388-395 (1993).
7. J.S. Arora, *Introduction to Optimum Design*, McGraw-Hill, New York, (1989).
8. O. Axelsson and V.A. Barker, *Finite Element Solution of Boundary Value Problems*, Academic Press, New York, (1984).
9. E.R. Walsh, *Methods of Optimization*, John Wiley and Sons, London, (1975).
10. J. Bialek and D.J. Grey, The application of clustering and factorisation tree techniques for the parallel solution of sparse network equations, *IEE, Proceedings C*, 609–616 (1994).
11. M.H.B.M. Shariff, A constrained element-by-element method for slightly compressible and incompressible materials and its parallel application for transputers, In *Advances in Parallel and Vector Processing for Structural Mechanics* (Edited by B.H.V. Topping and M. Papadrakakis), Civil-Comp Press, (1994).
12. R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, Society for Industrial and Applied Mathematics, Philadelphia, (1994).