Accelerated Methods for the Block Cimmino Row Projection Method for Solving Large Nonsymmetric Linear Systems

Emmanuel KAMGNIA
Department of Computer Science
University of Yaounde I
P.O. Box 812 Yaounde, CAMEROON

Abstract

The use of the row block projection method (block Cimmino) for solving nonsymmetric linear system $Ax = b$ gives rise to a symmetric positive definite system $Gx = g$, where B is the sum of orthogonal projection matrices $P_i = A_i(A_i^T A_i)^{-1} A_i^T$, and where the A_i^T's are block rows of A. In this paper we present an efficient implementation of the cgT method for solving the derived symmetric positive definite system. We use initial smoothing to accelerate the method and we derive efficient techniques for estimating the initial smoothed vector. Numerical results are given for the 2 row block case and since the eigenvalues of G can then be calculated, we by-pass the computation of orthogonal projections $P_i x$ of a vector x onto the range of A_i by solving a transformed equivalent system. Test results show that the method is robust.

Keywords: nonsymmtric linear system, projection methods, Cimmino row block projection methods, block Stiefel method, cgT method, smoothing, Krylov subspace, Lanczos method.

1 Introduction

Over the last few years, researchers have concentrated their efforts on developping efficient iterative solvers for nonsymmetric linear systems $Ax = b$, where A is large sparse and nonsingular. In general one can group these

solvers into four categories: matrix splitting, CG-like, residual polynomial and symmetrization methods. The first three categories are restricted to the case where $A+A^T$ is positive definite or the spectrum of A lie on one side of the imaginary axis. Symmetrization methods explicitely or implicitely transform the original system onto a symmetric positive definite system $Gx = g$ which can be solved using some the existing powerful methods designed for such systems. An example of the symmetrization method is to use the CG method on the normal equation $A^TAx = A^Tb$. This method, however, has very often hindered on the following observations: (i) forming A^TA can be costly in terms of storage and processing time, (ii) the condition number $\kappa(A^TA)$ is the square of $\kappa(A)$ which is a much serious problem especially when A is not well conditioned. A family of symmetrization methods which avoid these difficulties are the row projection methods. Let A be partitionned into N blocks as follows: $A^T = [A_1, A_2, \cdots, A_N]$ and b be partitionned conformally. A row projection method is any algorithm which requires the orthogonal projection $P_ix = A_i(A_i^TA_i)^{-1}A_i^Tx$ of a vector x onto range(A_i) to be computed. One such method is the block Cimmino method where the symmetric positive definite system is

$$(P_1 + P_2 + \cdots + P_N)x = g, \tag{1}$$

with

$$g = A_1\left(A_1^TA_1\right)^{-1}b_1 + A_2\left(A_2^TA_2\right)^{-1}b_2 + \cdots + A_N\left(A_N^TA_N\right)^{-1}b_N.$$

It is derived by premultiplying the system $Ax = b$ by

$$[A_1(A_1^TA_1)^{-1}, A_2(A_2^TA_2)^{-1}, \cdots, A_N(A_N^TA_N)^{-1}]$$

In the sequel we will write system (1) as

$$Gx = g. \tag{2}$$

g and the projection P_ix_k of a vector x_k onto the range of A_i could be formed by solving N independent linear least squares problems. For the 2 row block case, it can be shown [9], [10] that the eigenvalues of $G = P_1 + P_2$ are given by

$$\lambda_i(G) = 1 \pm c_i = 1 \pm \cos\theta_i,\ i = 1, 2, \cdots, n/2,$$

where $\theta_i's$ are the angles between the hyperplane generated by A_1 and A_2. For the 2 row block case, we could therefore study the behavior of a gradient

direction method for solving system (2) by solving instead of the equivalent system

$$(I - K)x = f, \tag{3}$$

where

$$K = diag(-c_1, -c_2, \cdots, -c_{n/2}, c_{n/2}, c_{n/2-1}, \cdots, c_1),$$

and where the $c_i's$, are defined above. This is the system that we later consider for our numerical experiments. It is worth mentioning that the block Cimmino method is well suited for parallel computation as the projection can be computed in parallel.

Our primary interest in this paper is to develop appropriate techniques to accelerate the row block Cimmino system. We consider two such techniques: the block Stiefel method[6] and the cgT method[4]. We show that the cgT method combined with initial smoothing is robust and we devise an efficient method for estimating the initial smoothed vector. Finely we show how a matrix-vector multiplication, a fundamental primitive in a cgT step, can be computed efficiently.

2 Block Stiefel Method[6]

The classical Stiefel iteration[4] for system (2) produces residuals r_i that satisfy

$$r_j = P_j(G)r_0, \tag{4}$$

where $P_j(\lambda)$ is a polynomial of degree $\leq j$ given by

$$P_j(\lambda) = \frac{T_j(\frac{\mu+\nu-2\lambda}{\mu-\nu})}{T_j(\frac{\mu+\nu}{\mu-\nu})}, \quad \nu \leq \lambda \leq \mu \tag{5}$$

in which $T_j(\xi)$ is the Tchebyshev polynomial of degree j given by

$$T_j(\xi) = \begin{cases} \cos(j \cos^{-1} \xi) & |\xi| \leq 1 \\ \\ \cosh(j \cosh^{-1} \xi) & |\xi| \geq 1. \end{cases} \tag{6}$$

If one has an estimate η of an interior eigenvalue λ_{s+1}, for a small integer s

$$0 < \nu \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_s < \eta \leq \lambda_{s+1} \leq \cdots \leq \lambda_m \leq \mu$$

and if

$$r_0 = \sum_{i=1}^{m} \tau_i z_i,$$

where the $z_i's$ are the eigenvector associated to the $\lambda_i's$, then

$$\begin{aligned} r_j &= P_j(G)r_0 \\ &= \sum_{i=1}^{s} \tau_i P_j(\lambda_i) z_i + \sum_{i=s+1}^{m} \tau_i P_j(\lambda_i) z_i \\ &= r'_j + r''_j. \end{aligned} \tag{7}$$

But

$$\begin{aligned} \frac{\|r''_j\|_2}{\|r''_0\|_2} &\leq \frac{1}{T_j(\frac{\mu+\eta}{\mu-\eta})}, \\ \frac{\|r'_j\|_2}{\|r'_0\|_2} &\leq \frac{T_j(\frac{\mu+\eta-2\lambda_1}{\mu-\eta})}{T_j(\frac{\mu+\eta}{\mu-\eta}).} \end{aligned}$$

Thus, r''_j is damped out faster than r'_j as j increases. The basic strategy of the block Stiefel method is to annihilate the contributions of the eigenvectors $z_1, z_2, \cdots, z_s$ in r_j so that $\|r_j\|_2 \longrightarrow 0$ as

$$\frac{1}{T_j(\frac{\lambda_m+\lambda_{s+1}}{\lambda_m-\lambda_{s+1}})},$$

and not as

$$\frac{1}{T_j(\frac{\lambda_m+\lambda_1}{\lambda_m-\lambda_1})}.$$

If $Z = (z_1, z_2, \cdots, z_s)$ is the matrix consisting of the s eigenvectors corresponding to s smallest eigenvalues of G, then instead of taking x_j as the j-th estimate, one takes

$$\hat{x}_j = x_j + Z(Z^T G Z)^{-1} Z^T r_j \tag{8}$$

for which $\hat{r}_j = g - G\hat{x}_j$ has a zero projection onto the space generated by $z_1, z_2, \cdots, z_s$. i.e., $Z^T \hat{r}_j = 0$. Since

$$Z^T G Z = diag(\lambda_1, \lambda_2, \cdots, \lambda_s),$$

we have

$$\hat{x}_j = x_j + \sum_{i=1}^{s} \frac{(z_i^T r_j)}{\lambda_i} z_i. \tag{9}$$

In practice the eigenvalues and eigenvectors of G are not known so that equation (9) cannot be used. We can however use methods such as cg to estimate the extreme eigenvalues λ_s and λ_m. A technique for approximating $Z(Z^TGZ)^{-1}Z^Tr_j$ is described in [6]. It is based on forming the QR factorization of the matrix

$$R_l = [r_{l-s+1}, r_{l-s+2}, \cdots, r_l],$$

where l is iteration number such that

$$\frac{1}{T_l(\frac{\lambda_m+\lambda_{s+1}}{\lambda_m-\lambda_{s+1}})} < \tau,$$

for a prescribed tolerance τ which indicates that r_j'' is sufficiently damped so that projection can be done.

3 The cgT Method[4]

Let us premultiply the system (2) by

$$B = G^{-1}[I - R(G)],$$

where $R(\lambda)$ is a polynomial such that $R(0) = 1$. We then obtain the system

$$[I - R(G)]x = G^{-1}[I - R(G)]g,$$

which we write as

$$Fx = f, \tag{10}$$

where

$$f = G^{-1}[I - R(G)]g. \tag{11}$$

Then

$$\lambda_i(F) = 1 - R(\lambda_i(G)),$$

and the process amounts to a spectral transformation of system (2), which, hopefully with a proper choice of $R(\lambda)$, will make the condition number of F become smaller than that of G. Solving system (10) is not straight forward because:

(i) to compute the initial residual one needs to form the right hand side f which would require solving the system $Gf = [I - R(G)]g$,

(ii) each matrix vector multiplication Fr_k will require forming the matrix polynomial vector product $R(G)r_k$ which may be expensive if the degree of $R(\lambda)$ is large.

The cgT Method is an efficient method for solving system (10) without forming f explicitly. The method has two stages:

(i) an inner stage, called inner method, where the Tchebychev method is used to form the matrix vector Fr_k;

(ii) an outer stage, called outer method, where the cg uses information generated by the inner method to update the approximate iterate.

The inner method is therefore subordinate to the outer method because for every step of the latter, a fixed number of steps of the inner method (m, say) must be carried out. $R(\lambda)$ can be chosen to be any residual polynomial $R_m(\lambda)$ satisfying $R_m(0) = 1$. For example Flanders and Shortley[4] propose the 3-term recurrence formula

$$\begin{aligned} R_{k+1}(\lambda) &= R_k(\lambda) + \Delta R_{k-1}(\lambda) \;\; k = 0, 1, 2, \cdots, m \\ \Delta R_k(\lambda) &= \frac{1}{\beta_k}[\lambda R_k(\lambda) + \alpha_{k-1}\Delta R_{k-1}(\lambda)], \end{aligned}$$

where $R_0 = 1$, $\Delta R_{-1}(\lambda) = 0$, for computing $R_m(\lambda)$. If β_k and α_k are computed as in algorithm 1 below, it can be shown that

$$R_k(\lambda) = \frac{T_k(\frac{\mu+\nu-2\lambda}{\mu-\nu})}{T_k(\frac{\mu+\nu}{\mu-\nu})}, \;\; \nu \leq \lambda \leq \mu, \tag{12}$$

where $T_k(\lambda)$ is defined as above. Let $R(\lambda)$ be the residual polynomial $R_m(\lambda)$ for a fixed m. In the sequel, we consider the conjugate direction methods for solving $Gx = g$ that use the following flow diagram

Algorithm 1:

1. set $\alpha_1 = 0$, $\Delta x_1 = 0$, $\Delta r_1 = 0$, $\omega = \cosh^{-1}\frac{\mu+\nu}{\mu-\nu}$

2. set k = 0

3. choose initial extimate x_0

4. form initial residual $r_0 = g - Gx_0$

5. repeat until convergence test met

5.1 compute

$$\beta_k = \begin{cases} \frac{\mu+\nu}{2} & k = 0 \\ \frac{\mu-\nu}{4} \frac{\cosh((k+1)\omega)}{\cosh(k\omega)} & k > 0 \end{cases}$$

5.2 compute

$$\Delta x_k = \frac{1}{\beta_k}(r_k + \alpha_{k-1}\Delta x_{k-1})$$

5.3 compute

$$\Delta r_k = \frac{1}{\beta_k}(Gr_k + \alpha_{k-1}\Delta r_{k-1})$$

5.4 set $x_{k+1} = x_k + \Delta x_k$

5.5 set $r_{k+1} = r_k + \Delta r_k$

5.6 if convergence tolerance met stop. Else,

5.7 compute

$$\alpha_k = \begin{cases} 0 & k < 0 \\ \frac{\mu-\nu}{4} \frac{\cosh(k\omega)}{\cosh((k+1)\omega)} & k \geq 0 \end{cases}$$

5.8 set k = k + 1

end repeat

To solve system (10), the cgT Method proceeds as follows:

3.1 Forming the Initial Residual $\tilde{r}_0 = f - F\tilde{x}_0$

Suppose we solve $Gx = g$, using the inner method and starting with $x_0 = \tilde{x}_0$. Then

$$r_0 = g - Gx_0.$$

Since

$$r_m = R_m(G)r_0,$$

it follows that

$$r_m = (I - F)r_0.$$

Thus,

$$r_0 - r_m = Fr_0,$$

so that

$$G(x_m - x_0) = Fg - FGx_0.$$

i.e.,

$$x_m - x_0 = G^{-1}Fg - Fx_0 = \tilde{r}_0.$$

Thus to form the initial residual $\tilde{r}_0 = f - F\tilde{x}_0$ we carry out m steps of the inner method for the system (2), using $\tilde{x}_0$ as the initial estimate, then set $\tilde{r}_0 = x_m - \tilde{x}_0$.

3.2 Computing the Matrix-vector Product $F\tilde{r}_k$

Suppose we solve the system $Gx = \tilde{r}_k$ using the inner method and the initial estimate $x_0 = 0$. Then $r_0 = \tilde{r}_k$. But

$$r_m = R_m(G)r_0 = R_m(G)\tilde{r}_k.$$

Therefore

$$r_0 - r_m = [I - R(G)]\tilde{r}_k.$$

Thus in order to compute $F\tilde{r}_k$, we take m steps of the inner method for the system $Gx = \tilde{r}_k$, using $x_0 = 0$ as the initial estimate, then set $F\tilde{r}_k = \tilde{r}_k - r_m$.

3.3 Initial Smoothing

To make the cgT Method more efficient, it is recommanded to eliminate the contribution of eigenvalues above a certain (not too small) limit σ, by first taking k steps of the Tchebychev method with $\nu = \sigma$ (in place of $\nu < \lambda_{min}$). For a large enough k, the residual vector and the error vector of the k-th approximate lies "almost" in the invariant subspace of a relatively small dimension l, spanned by the eigenvectors corresponding to the eigenvalues below σ. This process is called **smoothing**, and the k-th resulting approximate x_k is called the **smoothed vector.** We can then start the cgT Method using the smoothed vector x_k as the initial estimate.

Smoothing is useful only when k is large enough. But a large k make smoothing expensive. In what follows, we explore ways of reducing the cost of smoothing (measured in terms of matrix vector multiplication), by approximating the smoothed vector.

3.4 Method for Approximating the Smoothed Vector

Let x_k be the smoothed vector obtained after k steps of Tchebychev method. Since

$$r_k \equiv g - Gx_k = R_k(G)r_0$$

and

$$r_0 = g - Gx_0,$$

it follows that

$$\begin{aligned} r_k \equiv g - Gx_k &= R_k(G)r_0 \\ &= R_k(G)(g - Gx_0) \end{aligned}$$

$$\begin{aligned} \Rightarrow Gx_k &= g - R_k(G)g + R_k(G)Gx_0 \\ &= (I - R_k(G))g + R_k(G)Gx_0 \end{aligned}$$

$$\Rightarrow x_k = G^{-1}[I - R_k(G)]g + G^{-1}R_k(G)Gx_0. \tag{13}$$

Since $R_k(0) = 1$, it follows that

$$P(\lambda) = \frac{1}{\lambda}[1 - R_k(\lambda)]$$

is a polynomial of degree k-1 which we will denote by $P_{k-1}(\lambda)$. Then

$$x_k = P_{k-1}(G)g + R_k(G)x_0. \tag{14}$$

If $x_0 = 0$, then

$$x_k = P_{k-1}(G)g. \tag{15}$$

In this section we consider an approximation of (15) of the form

$$\bar{x}_k \equiv \gamma V P_{k-1}(T_m)e_1, \tag{16}$$

where γ is a scalar, V an $n \times m$ matrix, T_m a tridiagonal matrix of dimension m, all to be determined. Here, e_1 is the first unit vector. Hopefully, m will be much smaller than k so that the right hand side of (16) is much more cheaper than that of (15). We bigin by showing that the polynomials $P_k(\xi)$ satisfy a 3-term recurrence.

Theorem 1 *The polynomials*

$$P_k(\xi) = \frac{1}{\xi}[1 - R_{k+1}(\xi)]$$

staisfy the 3-term recurrence relationship,

$$\begin{aligned} P_k(\xi) &= 2\frac{T_k(\alpha)}{T_{k+1}(\alpha)}(\alpha - \beta\xi)P_{k-1}(\xi) \\ &- \frac{T_{k-1}(\alpha)}{T_{k+1}(\alpha)}P_{k-2}(\xi) + 2\beta\frac{T_k(\alpha)}{T_{k+1}(\alpha)}, \quad k \geq 2, \end{aligned} \tag{17}$$

where

$$\begin{aligned} P_0(\xi) &= \frac{\beta}{\alpha}, \\ P_1(\xi) &= -\frac{2\beta}{2\alpha^2 - 1}(\beta\xi - 2\alpha), \end{aligned}$$

with

$$\alpha = \frac{\mu + \nu}{\mu - \nu}, \quad \beta = \frac{2}{\mu - \nu}.$$

Proof: from (12)

$$R_{k+1}(\xi) = \frac{T_{k+1}(\alpha - \beta\xi)}{T_{k+1}(\alpha)} = \frac{T_{k+1}(\zeta)}{T_{k+1}(\alpha)},$$

where

$$\zeta = \alpha - \beta\xi.$$

From the properties of the Chebyshev polynomials $T_k(\zeta)$ we have,

$$T_{k+1}(\zeta) = 2\zeta T_k(\zeta) - T_{k-1}(\zeta), \quad -1 \leq \zeta \leq 1.$$

Thus

$$\begin{aligned} R_{k+1}(\xi) = \frac{T_{k+1}(\zeta)}{T_{k+1}(\alpha)} &= 2\zeta\frac{T_k(\alpha)}{T_{k+1}(\alpha)}\frac{T_k(\zeta)}{T_k(\alpha)} - \frac{T_{k-1}(\alpha)}{T_{k+1}(\alpha)}\frac{T_{k-1}(\zeta)}{T_{k-1}(\alpha)} \\ &= 2\zeta\frac{T_k(\alpha)}{T_{k+1}(\alpha)}R_k(\xi) - \frac{T_{k-1}(\alpha)}{T_{k+1}(\alpha)}R_{k-1}(\xi) \end{aligned}$$

i.e.

$$R_{k+1}(\xi) = 2\frac{T_k(\alpha)}{T_{k+1}(\alpha)}(\alpha - \beta\xi)R_k(\xi) - \frac{T_{k-1}(\alpha)}{T_{k+1}(\alpha)}R_{k-1}(\xi). \tag{18}$$

Now since

$$P_k(\xi) = \frac{1}{\xi}[1 - R_{k+1}(\xi)],$$

it follows that

$$\begin{aligned} 1 - \xi P_k(\xi) &= \frac{2\alpha T_k(\alpha) - T_{k-1}(\alpha)}{T_{k+1}(\alpha)} \\ &+ \xi\left(\frac{T_{k-1}(\alpha)}{T_{k+1}(\alpha)} P_{k-2}(\xi) - 2\frac{T_k(\alpha)}{T_{k+1}(\alpha)}\beta\right) \\ &- 2\frac{T_k(\alpha)}{T_{k+1}(\alpha)}\xi(\alpha - \beta\xi)P_{k-1}(\xi). \end{aligned}$$

Since $2\alpha T_k(\alpha) - T_{k-1}(\alpha) = T_{k+1}(\alpha)$, we have

$$\begin{aligned} P_k(\xi) &= 2\frac{T_k(\alpha)}{T_{k+1}(\alpha)}(\alpha - \beta\xi)P_{k-1}(\xi) \\ &- \frac{T_{k-1}(\alpha)}{T_{k+1}(\alpha)} P_{k-2}(\xi) + 2\beta\frac{T_k(\alpha)}{T_{k+1}(\alpha)}, \quad k \geq 2. \end{aligned}$$

This concludes the proof.

Returning to the approximation of x_k, we observe that

$$x_k \in span\{g, Gg, G^2g, \cdots, G^{k-1}g\},$$

the Krylov subspace of dimension k. We would like to seek an approximation from the Krylov subspace

$$K_m \equiv \{v, Gv, G^2v, \cdots, G^{m-1}v\},$$

for a small m. We choose the initial vector to be $v = g/\|g\|_2$ and generate the basis of K_m using the well known Lanczos algorithm which will produce an orthogonal basis

$$V_m = \{v_1, v_2, \cdots, v_m\}$$

of K_m such that

$$GV_m = V_m T_m + \delta v_{m+1} e_m^T, \tag{19}$$

where $V_m^T v_{m+1} = 0$, and T_m is a tridiagonal matrix of dimension m. Since V_m is orthogonal, $V_m V_m^T$ is a projection onto K_m and consequently,

$$\check{x}_k = V_m V_m^T x_k$$

is the projection of x_k onto K_m and as such the closest approximation of x_k from K_m. Since

$$V_m e_1 = \frac{g}{\|g\|_2} = v_1,$$

it follows that

$$\begin{aligned} \check{x}_k &= V_m V_m^T P_{k-1}(G) g \\ &= \|g\|_2 V_m V_m^T P_{k-1}(G) V_m e_1. \end{aligned}$$

Recalling (16), if we choose $\gamma = \|g\|_2$ and $V = V_m$, we therefore approximate $V_m^T P_{k-1}(G) V_m$ with $P_{k-1}(T_m)$. The following theorem will help to bound the the error $\|x_k - \bar{x}_k\|_2$.

Theorem 2 *Let $Q_{m-1}(\xi)$ be a polynomial of degree $\leq m-1$ that approximates $P_{k-1}(\xi)$ and let*

$$r_m = P_{k-1}(\xi) - Q_{m-1}(\xi). \tag{20}$$

Then,

$$\|P_{k-1}(G)g - \|g\|_2 V_m P_{k-1}(T_m) e_1\|_2 \leq \|g\|_2 (\|r_m(G)\|_2 + \|r_m(T_m)\|_2) \tag{21}$$

Proof: From equation (20) we have,

$$\begin{aligned} P_{k-1}(G)g &= r_m(G)g + Q_{m-1}(G)g \\ &= \|g\|_2 [Q_{m-1}(G)v_1 + r_m(G)v_1]. \end{aligned}$$

But from equation (19),

$$G^j v_1 = V_m T_m^j e_1, \quad j \leq m-1.$$

Thus

$$Q_{m-1}(G)v_1 = V_m Q_{m-1}(T_m) e_1.$$

Since

$$Q_{m-1}(T_m)e_1 = P_{k-1}(T_m)e_1 - r_m(T_m)e_1,$$

it folows that

$$P_{k-1}(G)g = \|g\|_2 [V_m P_{k-1}(T_m)e_1 - V_m r_m(T_m)e_1 + r_m(G)v_1].$$

Therefore

$$\|P_{k-1}(G)g - \|g\|_2 V_m P_{k-1}(T_m) e_1\|_2 \leq \|g\|_2 [\|r_m(G)\|_2 + \|r_m(T_m)\|_2],$$

which concludes the proof.

As a consequence of theorem (2), the error can be estimated by bounding each of the terms in the right hand side of (21). We establish a bound for the case when

$$Q_{m-1}(\xi) = P_{m-1}(\xi),$$

for which

$$r_m(\xi) = \frac{1}{\xi}[R_m(\xi) - R_k(\xi)].$$

Since

$$\max_{\nu \le \xi \le \mu} |R_k(\xi)| \approx 2 \left(\frac{\sqrt{\mu} + \sqrt{\nu}}{\sqrt{\mu} - \sqrt{\nu}} \right)^k, \tag{22}$$

and

$$\lambda_1(G) \le \lambda_1(T_m), \quad \lambda_m(T_m) \le \lambda_n(G),$$

and since $m \ll k$, it follows that

$$\|P_{k-1}(G)g - \|g\|_2 V_m P_{k-1}(T_m)e_1\|_2 \le \frac{4}{\nu} \left(\frac{\sqrt{\mu} + \sqrt{\nu}}{\sqrt{\mu} - \sqrt{\nu}} \right)^k. \tag{23}$$

This is a very crude bound and a sharper bound can be found. For example when $P_{k-1}(G)g - \|g\|_2 V_m P_{k-1}(T_m)e_1$ is expanded, it becomes apparent that $r_m(G)g$ is a polynomial of degree k-m in G and the power k in the right hand side of (23) is an overestimation.

Returning to the problem of approximating the smoothed vector, we now use

$$\bar{x}_k = \|g\|_2 V_m P_{k-1}(T_m)e_1 \tag{24}$$

to start the cgT method. One way of forming $\bar{x}_k$ is to use the 3-term recurrence relationship (17) which will cost k tridiagonal matrix vector multiplications. But as indicated by the equations (14) and (15), $P_{k-1}(T_m)e_1$ is the estimate obtained after k steps of Tchebyshev method for solving

$$T_m x = e_1. \tag{25}$$

Since m is relatively small compared to n, solving directely (25) not only is far cheaper than k tridiagonal matrix vector multiplications, but also gives a better value of $\bar{x}_k$. Our test results confirm this assertion. A better value for $\bar{x}_k$ is translated into a faster convergence.

4 Numerical Results, Summary and Comments

Numerical results reported here were conducted on a SUN work station while the author was visiting IRISA in Rennes and only deal with the 2 row block case. This is mainly because the eigenvalues of $P_1 + P_2$ can be expressed in terms the angles between the 2 subspaces generated by A_1 and A_2 and since the behavior of a gradient method in solving a linear system depends mainly on the distribution of the eigenvalues of the coefficient matrix, we can by-pass the computation of the projection in the block Cimmino system, by solving the equivalent diagonal system

$$(I - K)x = g, \tag{26}$$

where

$$K = diag(-c_1, -c_2, \cdots, -c_{n/2}, c_{n/2}, c_{n/2-1}, \cdots, c_1),$$

and where the $c_i's$, which are defined above, are generated randomly so as to obtain a desired distribution of eigenvalues of $P_1 + P_2$. Another advantage in dealing with a diagonal system is that we are able to handle large size systems. As long as the cost is measured in terms of matrix-vector multiplication, the behavior of a gradient direction method on both systems (2) and (26) should be identical. We have, however, carried out the full investigation using a) a matrix obtained from the finite difference discretization of a 2 dimensional elliptic partial differential equation with nonconstant coefficients, and b) a matrix generated experimentally. Tests were conducted on an IBM PS/PV 486DX2 of the department of computer science of the University of Yaounde I in Cameroon, but the small size of the systems used does not warrant publishing the results along side those obtained on a SUN work station.

In the experiments being reported here, systems (26) are constructed as follows: the $c_i's$ and the true solution vector x are first generated randomly then, the right hand side vector is obtained as $g = (I - K)x$. Throughout, we use $n = 10000$ and a relative error equal to 0.5E-05. Tables 1 and 2 display the results for the block Stiefel when the projection $Z(Z^TGZ)^{-1}Z^T$ is (i) computed exactly and (ii) approximated. In both cases, $s = 5$, $\tau = 0.5E - 02$ and the prescribed maximum number of iterations before projection is 200. The exact number of eigenvalues that lie below η is not known and the first column gives the number of eigenvalues missed. The block Stiefel under good conditions, i.e., good estimate of how many eigenvalues lie below ν and good approximation of their corresponding eigenvectors has performed well. However, tables (1,2) seems to

Table 1: block Stiefel method (exact projection used)

eig. missed	matvet mult	iter. before proj.(prec.)	iter. after proj.
0	409	181(0.5E-2)	228
50	254	43(0.5E-2)	211
100	252	30(0.5E-2)	222
500	252	15(0.5E-2)	237
1000	252	30(0.6487)	222

$\lambda_1 = 0.118153E-04 \leq \lambda_i \leq \lambda_n = 1.9999881847$

Table 2: block Stiefel method (projection approximated)

eig. missed	matvet mult	iter. before proj.(prec.)	iter. after proj.
0	533	103(0.5E-2)	430
50	468	40(0.5E-2)	428
100	447	30(0.5E-2)	417
500	436	13(0.5E-2)	426
1000	435	10(0.5E-2)	425

$\lambda_1 = 0.118153E-04 \leq \lambda_i \leq \lambda_n = 1.9999881847$

indicate that early projection may not be not too critical as long as the value of τ corresponding to the early projection step is small enough.

Tables 3 through 5 show the result for the cgT method. In each case the initial smoothing vector x_k is approximated using Lanczos method. The degrees of the Tchebychev polynomials used for the inner step are given in columns 2 to 4. The cgT method combined with initial smoothing seems to be the more promising approach, especially if $\bar{x}_k$ is used to approximate the smoothed vector. However, test results indicate that in the presence of a large number of small eigenvalues, one would gain more in using smoothing to eliminate the contribution of these small eigenvalues. If these small eigenvalues are not dampted out, the degree of the polynomial in the inner T-method of cgT should be kept small, thus making the cgT method behaves more like the cg method.

References

[1] Å Björck & G. Golub, *Numerical Method for Computing Angles Between Linear Subspaces*, Mathematics of Computation 27(123)(1973), 579-594.

Table 3: cgT method with $\lambda(smoothing) \equiv \lambda_{10}$

Number of matrix-vector multiplication				
# Lanczos step(deg. smmoting)	deg. T-poly.			$\frac{\|r_k\|_2}{\|r_0\|_2}$
	0	1	2	
10 (k=200)	358	355	359	0.5E-2
20 (k=200)	357	356	357	0.3E-2
30 (k=200)	355	352	351	0.2E-2
exact smoothing(k=100)	361	360	370	0.7E-3

$\lambda_1 = 0.118153E-04 \leq \lambda_i \leq \lambda_n = 1.9999881847$

Table 4: cgT method with $\lambda(smoothing) \equiv \lambda_1$

Number of matrix-vector multiplication				
# Lanczos step(deg. smmoting)	deg. T-poly.			$\frac{\|r_k\|_2}{\|r_0\|_2}$
	0	1	2	
10 (k=200)	337	338	340	0.19E-1
20 (k=200)	337	340	342	0.14E-1
30 (k=200)	340	340	343	0.10E-1
exact smoothing (k=100)	379	380	411	0.1471

$\lambda_1 = 0.244E-03 \leq \lambda_i \leq \lambda_n = 1.999756$

Table 5: cgT method with $\lambda(smoothing) \equiv \lambda_{100}$

Number of matrix-vector multiplication				
# Lanczos step(deg. smmoting)	deg. T-poly.			$\frac{\|r_k\|_2}{\|r_0\|_2}$
	0	1	2	
10 (k=200)	384	383	383	0.54E-2
20 (k=200)	384	384	383	0.12E-2
30 (k=200)	384	384	382	0.44E-5
exact smoothing (k=100)	385	386	388	0.25E-6

$\lambda_1 = 0.244E-03 \leq \lambda_i \leq \lambda_n = 1.999756$

[2] E. GALLOPOULOS & Y. SAAD, *Efficient Solution of Parabolic Equations by Polynomial Approximation Methods*, CSRD Technical Report no. 969, Center for Supercomputing Research & Development, University of Illinois at Urbana-Champaign.

[3] G. H. GOLUB & C. F. VAN LOAN, *Matrix Computations*, Second Edition, The John Hopkins University Press.

[4] M. ENGELI, TH. GINSI, H. RUTISHAUSER & E. STIEFEL, *Refined Iterative Methods for Computation of the Solution and Eigenvalues of Self-Adjoint Boundary Value Problems*, Mitteilungen aus dem Institut für angewandte Mathematik, 8(1959.

[5] O. G. JOHNSON, C. A. MICHELLI & G. PAUL, *Polynomial Preconditioners for Conjugate Gradient Calculation*, SIAM J. Numer. Anal., 2(20) (1983), 362,376.

[6] Y. SAAD, A. SAMEH & P. SAYLOR, *Solving Elliptic Difference Equations on Linear Array of Processors*, SIAM J. Sci. Stat. Comp., 4(6)(1985), 1049-1063.

[7] P. SAYLOR, *Leapfrog Variants of Iterative Methods for Linear Algebraic Equations*, Technical Report no. UIUCDCS-R-87-1373, Department of Computer Science, University of Illinois at Urbana-Champaign.

[8] R. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Inc. 1962.

[9] R. BRAMLEY, *Row projection methods for linear systems*, PhD Thesis 881(1989), Center for Supercomputer Research and Development, University of Illinois at Urbana Champaign, Urbana, IL.

[10] R. BRAMLEY AND A. SAMEH,*Row projection methods for large nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 13(1992), 168-193

[11] T. ELFVING, *Block-iterative for consistent and inconsistent linear equations*, Num. Math., 35(1980), 1-12

[12] C. KAMATH AND A. SAMEH, *A projection method for solving nonsymmetric linear systems on multiprocessors*, Parallel Computing, 9(1988), 291-312