SIAM J. SCI. COMPUT.
Vol. 21, No. 5, pp. 1909–1926
© 2000 Society for Industrial and Applied Mathematics

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

A DEFLATED VERSION OF THE CONJUGATE GRADIENT ALGORITHM*

Y. SAAD†, M. YEUNG‡, J. ERHEL§, AND F. GUYOMARC'H§

Abstract. We present a deflated version of the conjugate gradient algorithm for solving linear systems. The new algorithm can be useful in cases when a small number of eigenvalues of the iteration matrix are very close to the origin. It can also be useful when solving linear systems with multiple right-hand sides, since the eigenvalue information gathered from solving one linear system can be recycled for solving the next systems and then updated.

Key words. conjugate gradient, deflation, multiple right-hand sides, Lanczos algorithm

AMS subject classifications. 65F10, 65F15

PII. S1064829598339761

1. Introduction.

A number of recent articles have established the benefits of using eigenvalue deflation when solving nonsymmetric linear systems with Krylov subspace methods. It has been observed that significant improvements in convergence rates can be achieved from Krylov subspace methods by adding to these subspaces a few approximate eigenvectors associated with the eigenvalues closest to zero [2, 4, 7, 8, 13, 14]. In practice, approximations to the eigenvectors closest to zero are obtained from the use of a certain Krylov subspace; then these approximations are dynamically updated using the new Krylov subspace. Results of experiments obtained from these variations indicate that the improvement in convergence over standard Krylov subspaces of the same dimension can sometimes be substantial, especially when the convergence of the original scheme is hampered by a small number of eigenvalues near zero; see, e.g., [2, 8].

In this paper we consider extensions of this idea to the conjugate gradient (CG) algorithm for the symmetric case. Our starting point is an algorithm recently proposed by Erhel and Guyomarc'h [5]. This is an augmented subspace CG method aimed at linear systems with several right-hand sides. Erhel and Guyomarc'h [5] propose an algorithm which adds one specific vector obtained from a subspace related to a previous right-hand side. We first extend this algorithm to one which handles an arbitrary block W of vectors. We note that introducing an arbitrary W into the Krylov subspace of CG has already been considered by Nicolaides in [9]. The algorithm introduced in this paper is mathematically equivalent to the one in [9]. Nicolaides's algorithm is directly derived from a *deflated* Lanczos procedure and uses the 3-term recurrence version of the conjugate gradient algorithm. The algorithm in this paper exploits the link between the Lanczos algorithm and the standard CG algorithm. The LDL^T factorization of the projected system obtained from the same

*Received by the editors June 1, 1998; accepted for publication (in revised form) February 12, 1999; published electronically April 28, 2000. An extended abstract of this paper appeared in the Proceedings of the Fifth Copper Mountain Conference on Iterative Methods held in Colorado from March 30 to April 3, 1998.

http://www.siam.org/journals/sisc/21-5/33976.html

†University of Minnesota, Department of Computer Science and Engineering. The work of this author was supported by NSF grant CCR-9618827.

‡University of California at Irvine, Department of Mathematics, 420 Physical Sciences I, Irvine, CA 92697-3875 (myeung@math.uci.edu). The work of this author was supported by NSF grant CCR-9405380.

§Institut National de Recherche en Informatique et Automatique (INRIA), Rennes, France.

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

deflated Lanczos procedure leads to a procedure that is closer to the standard CG algorithm.

In the second part of the paper, we apply this technique to the situation when the block W of added vectors is a set of approximate eigenvectors. This, in turn, is used for solving linear systems with sequential multiple right-hand sides. This kind of system was also discussed in [15] for GMRES.

2. The deflated Lanczos algorithm. Consider a symmetric positive definite (SPD) matrix $A \in \mathbf{R}^{n\times n}$, and let k real vectors $w_1, w_2, \ldots, w_k$ be given, along with a unit vector v_1 that is orthogonal to w_i for $i = 1, 2, \ldots, k$. Define $W = [w_1, w_2, \ldots, w_k]$. We assume that $[w_1, w_2, \ldots, w_k]$ is a set of linearly independent vectors. Since A is SPD, the matrix $W^T AW$ is then nonsingular.

The deflated Lanczos algorithm builds a sequence $\{v_j\}_{j=1,2,\ldots}$ of vectors such that

$$v_{j+1} \perp span\{W, v_1, v_2, \ldots, v_j\} \tag{2.1}$$

and

$$\|v_{j+1}\|_2 = 1. \tag{2.2}$$

To obtain such a sequence, we apply the standard Lanczos procedure [12, p. 174] to the auxiliary matrix

$$B := A - AW\left(W^T AW\right)^{-1} W^T A \tag{2.3}$$

with the given initial v_1. The matrix B is symmetric but not necessarily positive definite. Then we have a sequence $\{v_j\}_{j=1,2,\ldots}$ of Lanczos vectors which satisfies

$$BV_j = V_j T_j + \sigma_{j+1} v_{j+1} e_j^T, \tag{2.4}$$

where

$$T_j = \begin{bmatrix} \rho_1 & \sigma_2 & & & \\ \sigma_2 & \rho_2 & \sigma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma_{j-1} & \rho_{j-1} & \sigma_j \\ & & & \sigma_j & \rho_j \end{bmatrix}$$

and where $V_j := [v_1, v_2, \ldots, v_j]$ and e_j is the last column of the $j \times j$ identity matrix I_j.

It is guaranteed by the Lanczos procedure that the vectors v_j are orthonormal to each other. From (2.4), it follows that

$$\sigma_{j+1} v_{j+1} = Bv_j - \sigma_j v_{j-1} - \rho_j v_j.$$

Using induction and noting that $W^T B = 0$, we have $v_{j+1}^T W = 0$ for $j = 1, 2, \ldots$, and hence the sequence $\{v_j\}_{j=1,2,\ldots}$ of Lanczos vectors has the properties (2.1) and (2.2).

Substituting B in the Lanczos algorithm applied to B with the right-hand side of (2.3) gives the following algorithm.

ALGORITHM 2.1. Deflated Lanczos algorithm.

1. *Choose k vectors $w_1, w_2, \ldots, w_k$. Define $W = [w_1, w_2, \ldots, w_k]$.*
2. *Choose an initial vector v_1 such that $v_1^T W = 0$ and $\|v_1\|_2 = 1$. Set $\sigma_1 v_0 = 0$.*
3. *For $j = 1, 2, \ldots, m$, Do:*

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

4. *Solve* $W^T AW\hat{\nu}_j = W^T Av_j$ *for* $\hat{\nu}$
5. $z_j = Av_j - AW\hat{\nu}_j$
6. $\rho_j = v_j^T z_j$
7. $\hat{v}_{j+1} = z_j - \sigma_j v_{j-1} - \rho_j v_j$
8. $\sigma_{j+1} = \|\hat{v}_{j+1}\|_2$; *If* $\sigma_{j+1} == 0$ *Exit.*
9. $v_{j+1} = \hat{v}_{j+1}/\sigma_{j+1}$
10. *EndDo*

The Deflated-CG algorithm proposed by Nicolaides [9] can be readily obtained from the above algorithm by deriving a sequence of iterates whose residual vectors are proportional to the v-vectors.

3. The Deflated-CG algorithm. We now turn to the linear system

$$Ax = b, \tag{3.1}$$

where A is SPD. Based on the deflated Lanczos procedure described in section 2, we wish to derive a projection method, following a standard technique used in the derivation of the CG algorithm [12, p. 179] from the standard Lanczos procedure.

Here, the w_i's and v_j's are as defined in the deflated Lanczos procedure and we use the notations W and V_j introduced in section 2. Assume an initial guess x_0 to (3.1) is given such that $r_0 := b - Ax_0 \perp W$. We set $v_1 = r_0/\|r_0\|_2$. Define

$$\mathcal{K}_{k,j}(A, W, r_0) \equiv span\{W, V_j\}.$$

At the jth step of our projection method, we seek an approximate solution x_j with

$$x_j \in x_0 + \mathcal{K}_{k,j}(A, W, r_0) \tag{3.2}$$

and

$$r_j = b - Ax_j \perp \mathcal{K}_{k,j}(A, W, r_0). \tag{3.3}$$

LEMMA 3.1. *If* x_j *and* r_j *satisfy* (3.2) *and* (3.3), *then*

$$r_j = c_j v_{j+1} \tag{3.4}$$

for some scalar c_j. *Thus* $\mathcal{K}_{k,j}(A, W, r_0) = span\{W, r_0, r_1, \ldots, r_{j-1}\}$ *and the residuals* r_j *are orthogonal to each other.*

Proof. Using (3.2), the approximate solution x_j can be written as

$$x_j = x_0 + W\hat{\xi}_j + V_j\hat{\eta}_j \tag{3.5}$$

for some $\hat{\xi}_j$ and $\hat{\eta}_j$. Moreover, from (2.3) and (2.4) we have

$$AV_j = AW\Delta_j + V_jT_j + \sigma_{j+1}v_{j+1}e_j^T,$$

where $\Delta_j := \left(W^T AW\right)^{-1} W^T AV_j$. Hence

$$\begin{aligned} r_j &= r_0 - AW\hat{\xi}_j - AV_j\hat{\eta}_j \\ &= r_0 - AW\hat{\xi}_j - \left(AW\Delta_j + V_jT_j + \sigma_{j+1}v_{j+1}e_j^T\right)\hat{\eta}_j. \end{aligned} \tag{3.6}$$

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Multiplying (3.6) with W^T, and using the orthogonality conditions (2.1) and (3.3), immediately leads to the following system of equations for $\hat{\xi}_j$ and $\hat{\eta}_j$:

$$W^T AW\hat{\xi}_j + W^T AW\Delta_j\hat{\eta}_j = 0.$$

Since $W^T AW$ is nonsingular, we get $\hat{\xi}_j = -\Delta_j\hat{\eta}_j$. Then (3.5) and (3.6) become

$$x_j = x_0 - W\Delta_j\hat{\eta}_j + V_j\hat{\eta}_j$$

and

$$r_j = r_0 - V_jT_j\hat{\eta}_j - \sigma_{j+1}v_{j+1}e_j^T\hat{\eta}_j. \tag{3.7}$$

Moreover, since $r_0 = \|r_0\|_2 v_1$ by definition and $V_j^T r_j = 0$ by (3.3), we have

$$V_j^T r_j = V_j^T r_0 - T_j\hat{\eta}_j = \|r_0\|_2 e_1 - T_j\hat{\eta}_j = 0,$$

where e_1 is the first column of I_j. Hence

$$\hat{\eta}_j = \|r_0\|_2 T_j^{-1} e_1. \tag{3.8}$$

Substituting (3.8) into (3.7), one can see that $r_j = c_j v_{j+1}$ for some scalar c_j. ∎

Let $T_j = L_jD_jL_j^T$ be the LDL^T decomposition of the symmetric matrix T_j. Define

$$P_j \equiv [p_0, p_1, \ldots, p_{j-1}] = (-W\Delta_j + V_j)\, L_j^{-T}\Lambda_j, \tag{3.9}$$

where $\Lambda_j = \mathrm{diag}\{c_0, c_1, \ldots, c_{j-1}\}$.

PROPOSITION 3.2. *The solution x_j, the residual r_j, and the descent direction p_j satisfy the recurrence relations*

$$\begin{aligned} x_j &= x_{j-1} + \alpha_{j-1}p_{j-1}, \\ r_j &= r_{j-1} - \alpha_{j-1}Ap_{j-1}, \\ p_j &= r_j + \beta_{j-1}p_{j-1} - W\hat{\mu}_j \end{aligned} \tag{3.10}$$

for some $\alpha_{j-1}, \beta_{j-1}, \hat{\mu}_j$. Thus $\mathcal{K}_{k,j}(A, W, r_0) = span\{W, p_0, p_1, \ldots, p_{j-1}\}$.

Proof. Let

$$\hat{\zeta}_j = \|r_0\|_2 \left(L_jD_j\Lambda_j\right)^{-1} e_1.$$

The approximate solution is then given by

$$\begin{aligned} x_j &= x_0 + \|r_0\|_2 \left(-W\Delta_j + V_j\right) T_j^{-1} e_1 \\ &= x_0 + P_j\hat{\zeta}_j. \end{aligned}$$

Because of the lower triangular structure of $L_jD_j\Lambda_j$, we have the relation

$$\hat{\zeta}_j = \begin{bmatrix} \hat{\zeta}_{j-1} \\ \alpha_{j-1} \end{bmatrix}$$

for some scalar α_{j-1}. The recurrence relation for x_j immediately follows.

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Now, rewriting (3.9) as

$$P_j \Lambda_j^{-1} L_j^T \Lambda_j = -W\Delta_j \Lambda_j + V_j \Lambda_j$$

and noting that $\Lambda_j^{-1} L_j^T \Lambda_j$ is unit upper bidiagonal, we have by comparing the last columns of both sides of the above equation

$$p_{j-1} - \beta_{j-2} p_{j-2} = -W\hat{\mu}_{j-1} + c_{j-1} v_j,$$

where $\beta_{j-2} = -c_{j-1} u_{j-1,j}/c_{j-2}$ and $\hat{\mu}_{j-1} = c_{j-1}\hat{\nu}_j$. Thus, the recurrence for the p_j's is obtained. ∎

PROPOSITION 3.3. *The vectors p_j are A-orthogonal to each other, i.e., $P_j^T A P_j$ is diagonal. In addition, they are also A-orthogonal to all w_i's, i.e., $W^T A P_j = 0$.*

Proof. Indeed,

$$\begin{aligned} P_j^T A P_j &= P_j^T \left(-AW\Delta_j + AV_j\right) L_j^{-T} \Lambda_j \\ &= P_j^T \left(V_j T_j + \sigma_{j+1} v_{j+1} e_j^T\right) L_j^{-T} \Lambda_j \\ &= \Lambda_j^T L_j^{-1} \left(-W\Delta_j + V_j\right)^T \left(V_j T_j + \sigma_{j+1} v_{j+1} e_j^T\right) L_j^{-T} \Lambda_j \\ &= \Lambda_j^T L_j^{-1} T_j L_j^{-T} \Lambda_j \\ &= \Lambda_j^T D_j \Lambda_j \end{aligned}$$

is diagonal and

$$\begin{aligned} W^T A P_j &= W^T \left(-AW\Delta_j + AV_j\right) L_j^{-T} \Lambda_j \\ &= W^T \left(V_j T_j + \sigma_{j+1} v_{j+1} e_j^T\right) L_j^{-T} \Lambda_j \\ &= 0. \qquad ∎ \end{aligned}$$

By using the orthogonality of r_j's and the A-orthogonality of p_j's, the coefficients in (3.10) can be expressed via the vectors W, r_j and p_j.

PROPOSITION 3.4. *The coefficients in Deflated-CG satisfy the relations*

$$\begin{aligned} \alpha_j &= \frac{r_j^T r_j}{p_j^T A p_j}, \\ \hat{\mu}_j &= \left(W^T A W\right)^{-1} W^T A r_j, \\ \beta_j &= \frac{r_{j+1}^T r_{j+1}}{r_j^T r_j}. \end{aligned} \tag{3.11}$$

Proof. Multiplying (3.10) with r_{j-1}^T yields

$$\alpha_{j-1} = \frac{r_{j-1}^T r_{j-1}}{r_{j-1}^T A p_{j-1}} = \frac{r_{j-1}^T r_{j-1}}{\left(\beta_{j-2} p_{j-2} + r_{j-1} - W\hat{\mu}_{j-1}\right)^T A p_{j-1}} = \frac{r_{j-1}^T r_{j-1}}{p_{j-1}^T A p_{j-1}}.$$

Similarly, the expressions for $\hat{\mu}_j$ and β_{j-1} are obtained as follows by applying $(AW)^T$ and $(Ap_{j-1})^T$ to (3.10), respectively,

$$\hat{\mu}_j = \left(W^T A W\right)^{-1} W^T A r_j,$$

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

$$\beta_{j-1} = -\frac{p_{j-1}^T A r_j}{p_{j-1}^T A p_{j-1}} = -\frac{1}{\alpha_{j-1}} \frac{(r_{j-1} - r_j)^T r_j}{p_{j-1}^T A p_{j-1}}$$
$$= \frac{1}{\alpha_{j-1}} \frac{r_j^T r_j}{p_{j-1}^T A p_{j-1}} = \frac{r_j^T r_j}{r_{j-1}^T r_{j-1}}. \qquad \square$$

Putting relations (3.10) and (3.11) together gives the following algorithm.

ALGORITHM 3.5. Deflated-CG.

1. *Choose k linearly independent vectors $w_1, w_2, \ldots, w_k$. Define $W = [w_1, w_2, \ldots, w_k]$.*
2. *Choose an initial guess x_0 such that $W^T r_0 = 0$, where $r_0 = b - Ax_0$.*
3. *Solve $W^T AW\hat{\mu}_0 = W^T Ar_0$ for $\hat{\mu}$ and set $p_0 = r_0 - W\hat{\mu}_0$.*
4. *For $j = 1, 2, \ldots, m$, Do:*
5. $\alpha_{j-1} = r_{j-1}^T r_{j-1} / p_{j-1}^T A p_{j-1}$
6. $x_j = x_{j-1} + \alpha_{j-1} p_{j-1}$
7. $r_j = r_{j-1} - \alpha_{j-1} A p_{j-1}$
8. ¿$\beta_{j-1} = r_j^T r_j / r_{j-1}^T r_{j-1}$
9. *Solve $W^T AW\hat{\mu}_j = W^T Ar_j$ for $\hat{\mu}$*
10. $p_j = \beta_{j-1} p_{j-1} + r_j - W\hat{\mu}_j$
11. *EndDo*

To guarantee that the initial guess x_0 satisfies $W^T r_0 = 0$, we can choose x_0 in the form

$$x_0 = x_{-1} + W\left(W^T AW\right)^{-1} W^T r_{-1}, \tag{3.12}$$

where x_{-1} is arbitrary and $r_{-1} := b - Ax_{-1}$. In fact, x_0 with $W^T r_0 = 0$ must have the form (3.12). To see this, we write $x_0 = x_{-1} + W\left(W^T AW\right)^{-1} W^T b$ for some x_{-1}. Since $W^T r_0 = 0$, we have $W^T Ax_{-1} = 0$ and hence $x_0 = x_{-1} + W\left(W^T AW\right)^{-1} W^T r_{-1}$. Formula (3.12) was also used in [5, 10, 11, 16].

A preconditioned version of Deflated-CG can be derived in a straightforward way. Suppose we are solving the split-preconditioned system

$$L^{-1} AL^{-T} y = L^{-1} b, \qquad x = L^{-T} y.$$

Set $M = LL^T$. Directly applying the Deflated-CG algorithm to the system $L^{-1} AL^{-T} y = L^{-1} b$ for the y-variable and then redefining the variables,

$$\begin{array}{lllllll} L^{-T} W & \rightarrow & W, & \qquad & Lr_j & \rightarrow & r_j, \\ L^{-T} y_j & \rightarrow & x_j, & & L^{-T} p_j & \rightarrow & p_j, \end{array} \tag{3.13}$$

yields the following algorithm.

ALGORITHM 3.6. Preconditioned Deflated-CG.

1. *Choose k linearly independent vectors $w_1, w_2, \ldots, w_k$. Define $W = [w_1, w_2, \ldots, w_k]$.*
2. *Choose x_0 such that $W^T r_0 = 0$, where $r_0 = b - Ax_0$. Compute $z_0 = M^{-1} r_0$.*
3. *Solve $W^T AW\hat{\mu}_0 = W^T Az_0$ for $\hat{\mu}$ and set $p_0 = -W\hat{\mu}_0 + z_0$.*
4. *For $j = 1, 2, \ldots, m$, Do:*
5. $\alpha_{j-1} = r_{j-1}^T z_{j-1} / p_{j-1}^T A p_{j-1}$
6. $x_j = x_{j-1} + \alpha_{j-1} p_{j-1}$
7. $r_j = r_{j-1} - \alpha_{j-1} A p_{j-1}$
8. $z_j = M^{-1} r_j$
9. $\beta_{j-1} = r_j^T z_j / r_{j-1}^T z_{j-1}$

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

10. *Solve* $W^T AW\hat{\mu}_j = W^T Az_j$ *for* $\hat{\mu}$
11. $p_j = \beta_{j-1}p_{j-1} + z_j - W\hat{\mu}_j$
12. *EndDo*

When W is a null matrix, Algorithm 3.6 reduces to the standard preconditioned CG algorithm; see, for instance, [12, p. 247]. In addition to the matrices A and M, five vectors and three matrices of storage are required: p, Ap, r, x, z, W, AW, and $W^T AW$.

4. Theoretical considerations. We observe that Deflated-CG is a generalization of AugCG [5] to any subspace W. Since the theory developed in [5] did not use the fact that W was a Krylov subspace, the convergence behavior of the Deflated-CG algorithm can be analyzed by exploiting the same theory. Define

$$H = I - W\left(W^T AW\right)^{-1}(AW)^T$$

to be the matrix of the A-orthogonal projection onto $W^{\perp_A}$ and

$$H^T = I - AW\left(W^T AW\right)^{-1}W^T$$

to be the matrix of the A^{-1}-orthogonal projection onto $W^{\perp}$. These matrices satisfy the equality

$$AH = H^T A = H^T AH. \tag{4.1}$$

We first prove that the Deflated-CG algorithm does not break down, and it converges. Then we derive a result on the convergence rate and some other properties.

PROPOSITION 4.1. *Algorithm* 3.5 *is equivalent to the balanced projection method* [6]*, defined by the solution space condition*

$$x_{j+1} - x_j \in \mathcal{K}_{k,j}(A, W, r_0), \tag{4.2}$$

and the Petrov–Galerkin condition

$$r_0 \perp W \quad \textit{and} \quad r_j \perp \mathcal{K}_{k,j}(A, W, r_0). \tag{4.3}$$

Proof. It is easy to show by induction that $p_j \in \mathcal{K}_{k,j}(A, W, r_0)$ hence the solution space condition is satisfied in Algorithm 3.5. It satisfies by construction the Petrov–Galerkin condition.

Conversely, the BPM defined with (4.2) and (4.3) satisfies all the recurrence relations stated in the Deflated-CG algorithm. ∎

The following result follows immediately.

THEOREM 4.2. *Let A be a symmetric positive definite matrix and W be a set of linearly independent vectors. Let x^* be the exact solution of the linear system $Ax = b$. The algorithm Deflated-CG applied to the linear system $Ax = b$ will not break down at any step. The approximate solution x_j is the unique minimizer of the error norm $\|x_j - x^*\|_A$ over the affine solution space $x_0 + \mathcal{K}_{k,j}(A, W, r_0)$ and there exists $\epsilon > 0$, independent of x_0, such that for all k*

$$\|x_j - x^*\|_A \leq (1 - \epsilon)\ \|x_{j-1} - x^*\|_A.$$

Proof. See Theorems 2.4, 2.6, and 2.7 in [6] and Theorem 2.2 in [5]. ∎

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

We can now directly apply the polynomial formalism built in [5] to obtain convergence properties.

THEOREM 4.3. *Let κ be the condition number of H^TAH. Then*

$$\|x_* - x_j\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^j \|x_* - x_0\|_A. \tag{4.4}$$

Proof. See Theorem 3.3 and Corollary 3.1 in [5]. Theorem 3.3 proves that $r_j = P_j(AH)r_0$, where P_j is a polynomial of degree j so that, using (4.1), we get $r_j = P_j(H^TAH)r_0$. Then the minimization property leads to the desired result. ∎

Deflated-CG can also be viewed as a preconditioned conjugate gradient (PCG) but with a singular preconditioner. Define

$$C = HH^T.$$

We use the taxonomy defined in [1], where two versions of PCG are denoted by $Omin(A, C, A)$ and $Odir(A, C, A)$.

LEMMA 4.4. *The Deflated-CG algorithm is equivalent to the version $Omin(A, C, A)$ of PCG applied to A with the preconditioner $C = HH^T$ and started with r_0 such that $r_0 \perp W$.*

Proof. The relations $p_0 = -W\hat{\mu}_0 + r_0$ and $p_j = -W\hat{\mu}_j + \beta_{j-1}p_{j-1} + r_j$ can be rewritten, respectively, as $p_0 = Hr_0$ and $p_j = Hr_j + \beta_{j-1}p_{j-1}$. However, since $r_j \perp W$, we have $r_j = H^Tr_j$ so,

$$p_0 = Cr_0 \quad \text{and} \quad p_j = Cr_j + \beta_{j-1}p_{j-1}.$$

These are exactly the same relations as those of $Omin(A, C, A)$. We must now prove that the coefficients α_j and β_j are the same. It is sufficient to show that $(r_j, Cr_j) = (r_j, r_j)$. We have $(r_j, Cr_j) = (r_j, HH^Tr_j) = (H^Tr_j, H^Tr_j) = (r_j, r_j)$. ∎

Here the preconditioner C is singular so that convergence is not guaranteed. However, since the initial residual is orthogonal to W, the following result can be proved.

THEOREM 4.5. *Deflated-CG is equivalent to the version $Odir(A, C, A)$ of PCG applied to A and C and started with $r_0 \perp W$. Therefore Deflated-CG converges.*

Proof. Since $\alpha_j = (r_j, r_j)$, we have $\alpha_j \neq 0$ except the case when x_j is the exact solution. Hence Deflated-CG does not break down and, as shown in [1], both versions *Omin* and *Odir* are equivalent. Now, using Theorem 3.1 in [1], we infer that Deflated-CG converges. ∎

To derive the convergence rate, we prove another equivalence.

THEOREM 4.6. *Deflated-CG is equivalent to CG version Omin (A,I,A), applied to the linear system $H^TAH\tilde{x} = H^Tb$. Therefore, the convergence rate is governed by the condition number κ of H^TAH and given by* (4.4).

Proof. Deflated-CG starts with $x_0 = H\tilde{x}_0 + Wy_0$ given by formula (3.12). Therefore,

$$r_0 = H^Tr_0 = H^T(b - Ax_0) = H^Tb - H^TAH\tilde{x}_0 - H^TAWy_0 = H^Tb - H^TAH\tilde{x}_0.$$

Algorithm Omin(A,I,A) applied to $H^TAH\tilde{x} = H^Tb$ is as follows:

1. Choose $\tilde{x}_0$. Define $\tilde{r}_0 = H^Tb - H^TAH\tilde{x}_0$ and $\tilde{p}_0 = \tilde{r}_0$.
2. For $j = 1, 2, \ldots,$ until convergence Do:

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3. $\tilde{\alpha}_{j-1} = \tilde{r}_{j-1}^T \tilde{r}_{j-1} / \tilde{p}_{j-1}^T H^T A H \tilde{p}_{j-1}$;
4. $\tilde{x}_j = \tilde{x}_{j-1} + \tilde{\alpha}_{j-1} \tilde{p}_{j-1}$;
5. $\tilde{r}_j = \tilde{r}_{j-1} - \tilde{\alpha}_{j-1} H^T A H \tilde{p}_{j-1}$;
6. $\tilde{\beta}_{j-1} = \tilde{r}_j^T \tilde{r}_j / \tilde{r}_{j-1}^T \tilde{r}_{j-1}$;
7. $\tilde{p}_j = \tilde{r}_j + \beta_{j-1} \tilde{p}_{j-1}$;
8. EndDo

Now, define $x_j = H\tilde{x_j} + W y_0$ and $p_j = H\tilde{p_j}$. Using $r_j = H^T r_j$ and $H^T A W y_0 = 0$, we get

$$r_j = H^T(b - Ax_j) = H^T b - H^T A H \tilde{x}_j = \tilde{r_j}.$$

Now

$$\tilde{p}_{j-1}^T H^T A H \tilde{p}_{j-1} = (H\tilde{p}_{j-1})^T A (H\tilde{p}_{j-1}) = p_j^T A p_j.$$

Putting everything together, we get $\tilde{\alpha}_j = r_j^T r_j / p_j^T A p_j$ and $\tilde{\beta}_{j-1} = r_j^T r_j / r_{j-1}^T r_{j-1}$. If we rewrite the above algorithm using x_j, r_j, p_j we get exactly algorithm Deflated-CG. Therefore, the classical result on the convergence rate of CG can be applied; see, e.g., [12, p. 194]. ∎

5. Systems with multiple dependent right-hand sides. In this section, we present a method which applies the Deflated-CG algorithm to the solution of several symmetric linear systems of the form

$$Ax^{(s)} = b^{(s)}, \qquad s = 1, 2, \ldots, \nu, \tag{5.1}$$

where $A \in \mathbf{R}^{n \times n}$ is SPD and where the different right-hand sides $b^{(s)}$ depend on the solutions of their previous systems. This problem has been recently considered by Erhel and Guyomarc'h [5]. The main idea in [5] is to solve the first system by CG and to recycle the Krylov subspace $K_m(A, x_0)$ created in the first system to accelerate the convergence in solving the subsequent systems. The disadvantage of this approach is that the memory requirements could be huge when m is large [3] since it requires keeping a basis of an earlier Krylov subspace $K_m(A, x_0)$. An alternative whose goal is to maintain a similar convergence rate, is to adopt the idea of eigenvalue deflation, as used in the Deflated-GMRES algorithm; e.g., see [2, 4, 8]. Deflated GMRES injects a few approximate eigenvectors into its Krylov solution subspace. These approximate eigenvectors are usually selected to be those hampering the convergence of the original scheme. Imitating the approach of Deflated-GMRES, we will add some approximate eigenvectors, usually corresponding to eigenvalues nearest zero, to the Krylov solution subspace when we solve each, except the first, system of (5.1). The eigenvectors are refined with each new system (5.1) being solved. In this way, we may expect that the convergence will be improved as more systems are solved. The memory requirements for this approach are fairly low, since only a small number of eigenvectors, typically 4 to 10, are required.

We start by deriving the convergence rate from section 4 when W is a set of eigenvectors. We label all the eigenvalues of A in increasing order:

$$\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n.$$

In the special case where the column vectors, $w_1, w_2, \ldots, w_k$, of W are exact eigenvectors of A associated with the smallest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$, then clearly

$$\kappa(H^T A H) = \lambda_n / \lambda_{k+1}.$$

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

In this special case, the improvement on the condition number of the equivalent system solved by Deflated-CG is explicitly known. When the column vectors of W are not exact but near eigenvectors associated with $\lambda_1, \lambda_2, \ldots, \lambda_k$, one can only expect that

$$\kappa(H^T AH) \approx \lambda_n/\lambda_{k+1}.$$

5.1. Computing approximate eigenvectors. There are several ways in which approximate eigenvectors can be extracted and improved from the data generated by the successive conjugate gradient runs applied to the systems of (5.1). Let

$$W^{(s)} = [w_1^{(s)}, w_2^{(s)}, \ldots, w_k^{(s)}]$$

be the desired set of eigenvectors to be used for the sth system. Initially $W^{(1)} = \emptyset$. In what follows vector and scalar quantities generated by Deflated-CG applied to the sth system of (5.1) are denoted by the superscript s. Ideally, $W^{(s)}$ is the set of eigenvectors associated with the k eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$, of A. Deflated-CG is used to solve each of the systems of (5.1) except the first which is solved by the standard CG.

After the sth system of (5.1) is solved, we update the set $W^{(s)}$ of approximate eigenvectors to be used for the next right-hand side, yielding a new system $W^{(s+1)}$. In [2], three projection techniques are described to obtain such approximations. Here we only describe one of them, suggested by Morgan [8] and referred to as *harmonic projection*. This approach yielded the best results in finding eigenvalues nearest zero. Given l linearly independent vectors $\{ z_1, z_2, \ldots, z_l \}$, the method computes the k desired approximate eigenvectors by solving the generalized eigenproblem

$$Gy - \theta Fy = 0,$$

where $Z := [z_1, z_2, \ldots, z_l]\,, G = (AZ)^T AZ$ and $F = Z^T AZ$. For each new system, the matrix Z is defined as follows:

$$Z^{(s)} = [W^{(s)}, P_l^{(s)}], \tag{5.2}$$

where[1]

$$P_l^{(s)} = \left[p_0^{(s)}, p_1^{(s)}, \ldots, p_{l-1}^{(s)} \right].$$

The generalized eigenvalue problem

$$G^{(s)} y_i - \theta_i F^{(s)} y_i = 0, \quad i = 1, \ldots, k, \tag{5.3}$$

is solved, where

$$F^{(s)} = \left(Z^{(s)}\right)^T AZ^{(s)}, \quad G^{(s)} = \left(AZ^{(s)}\right)^T AZ^{(s)} \tag{5.4}$$

and where $\theta_1, \ldots, \theta_k$ are the k smallest Ritz values. Then the new approximate set of eigenvectors to be used for the next system is defined as

$$W^{(s+1)} = Z^{(s)} Y^{(s)}, \tag{5.5}$$

[1]One may keep the system of residual vectors $r_i^{(s)}$ instead of the search directions $p_i^{(s)}$. But the formulas (5.7)–(5.9) become more complex.

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

where

$$Y^{(s)} \equiv [y_1^{(s)}, \ldots, y_k^{(s)}].$$

This describes the method mathematically. From the implementation point of view, it is possible to avoid the matrix-matrix multiplication $AZ^{(s)}$ in $G^{(s)}$ and $F^{(s)}$ in (5.4).

LEMMA 5.1. *Let*

$$R_l^{(s)} = \left[r_0^{(s)}, r_1^{(s)}, \ldots, r_{l-1}^{(s)} \right], \qquad \tilde{\Delta}_{l+1}^{(s)} = \left[\hat{\mu}_0^{(s)}, \hat{\mu}_1^{(s)}, \ldots, \hat{\mu}_l^{(s)} \right],$$

and

$$\tilde{L}_l^{(s)} = \begin{bmatrix} \frac{1}{\alpha_0^{(s)}} & & & \\ -\frac{1}{\alpha_0^{(s)}} & \frac{1}{\alpha_1^{(s)}} & & \\ & -\frac{1}{\alpha_1^{(s)}} & \ddots & \\ & & \ddots & \frac{1}{\alpha_{l-1}^{(s)}} \\ & & & -\frac{1}{\alpha_{l-1}^{(s)}} \end{bmatrix}, \quad \tilde{U}_l^{(s)} = \begin{bmatrix} 1 & -\beta_0^{(s)} & & & \\ & 1 & -\beta_1^{(s)} & & \\ & & \ddots & \ddots & \\ & & & 1 & -\beta_{l-1}^{(s)} \\ & & & & 1 \end{bmatrix}.$$

Then the matrix $AP_l^{(s)}$ can be computed by

$$AP_l^{(s)} = R_{l+1}^{(s)} \tilde{L}_l^{(s)} = \left(W^{(s)} \tilde{\Delta}_{l+1}^{(s)} + P_{l+1}^{(s)} \tilde{U}_l^{(s)} \right) \tilde{L}_l^{(s)}. \tag{5.6}$$

Proof. The result follows immediately from the following two relations of Algorithm 3.5:

$$r_j^{(s)} = p_j^{(s)} - \beta_{j-1}^{(s)} p_{j-1}^{(s)} + W^{(s)} \hat{\mu}_j^{(s)},$$

$$Ap_j^{(s)} = \frac{1}{\alpha_j^{(s)}} r_j^{(s)} - \frac{1}{\alpha_j^{(s)}} r_{j+1}^{(s)}. \qquad \square$$

Next, the relation established in the following proposition enables us to solve the harmonic problem (5.3) without requiring additional products with the matrix A.

THEOREM 5.2. *Define*

$$\tilde{D}_l^{(s)} = \mathrm{diag}\{ d_0^{(s)}, d_1^{(s)}, \ldots, d_{l-1}^{(s)} \}, \quad \textit{where} \quad d_j^{(s)} = \left(p_j^{(s)} \right)^T A p_j^{(s)},$$

and

$$\tilde{G}^{(s)} = \begin{bmatrix} \frac{d_0^{(s)}}{\alpha_0^{(s)}} \left(1 + \beta_0^{(s)} \right) & -\frac{d_1^{(s)}}{\alpha_0^{(s)}} & & & \\ -\frac{d_1^{(s)}}{\alpha_0^{(s)}} & \frac{d_1^{(s)}}{\alpha_1^{(s)}} \left(1 + \beta_1^{(s)} \right) & -\frac{d_2^{(s)}}{\alpha_1^{(s)}} & & \\ & -\frac{d_2^{(s)}}{\alpha_1^{(s)}} & \ddots & \ddots & \\ & & \ddots & \ddots & -\frac{d_{l-1}^{(s)}}{\alpha_{l-2}^{(s)}} \\ & & & -\frac{d_{l-1}^{(s)}}{\alpha_{l-2}^{(s)}} & \frac{d_{l-1}^{(s)}}{\alpha_{l-1}^{(s)}} \left(1 + \beta_{l-1}^{(s)} \right) \end{bmatrix}.$$

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Then the matrices $G^{(s)}$, $F^{(s)}$, and $AW^{(s)}$ are given by

$$G^{(s)} = \begin{bmatrix} \left(AW^{(s)}\right)^T AW^{(s)} & \left(W^{(s)}\right)^T AW^{(s)} \tilde{\Delta}_{l+1}^{(s)} \tilde{L}_l^{(s)} \\ \left(\tilde{\Delta}_{l+1}^{(s)} \tilde{L}_l^{(s)}\right)^T \left(W^{(s)}\right)^T AW^{(s)} & \tilde{G}^{(s)} \end{bmatrix}, \tag{5.7}$$

$$F^{(s)} = \begin{pmatrix} \left(W^{(s)}\right)^T AW^{(s)} & 0 \\ 0 & \tilde{D}_l^{(s)} \end{pmatrix}, \tag{5.8}$$

$$AW^{(s+1)} = [AW^{(s)}, AP_l^{(s)}]Y^{(s)}, \tag{5.9}$$

where $AP_l^{(s)}$ is given by (5.6).

Proof. The desired results can be obtained by using (5.5), (5.4), (5.6), and the A-othorgonality of P against W. ∎

In order to use this approach we must save $d_i^{(s)}, \alpha_i^{(s)}, \beta_i^{(s)}, \hat{\mu}_i^{(s)}$, and $p_i^{(s)}$ of the first l steps of the algorithm as well as the matrices $W^{(s)}$, $AW^{(s)}$, and $(W^{(s)})^T AW^{(s)}$.

5.2. Deflated-CG algorithm for dependent multiple right-hand sides. In summary the deflated algorithm for multiple right-hand sides works as follows. Assume we want to deflate with k eigenvectors. In a first run, the standard CG is run with a number of steps l which is no less than k. The data $d_i^{(1)}$, $\alpha_i^{(1)}$, $\beta_i^{(1)}$ for $i = 0, 1, \ldots, l-1$, and $P_l^{(1)} = [p_0^{(1)}, \ldots, p_{l-1}^{(1)}]$ are saved. Then $G^{(1)}$ and $F^{(1)}$ are computed according to (5.7) and (5.8) with $s = 1$. The eigenvalue problem (5.3) is solved for k eigenvectors which will constitute the columns of $Y^{(1)}$ and $W^{(2)}$ is computed as $W^{(2)} = [P_l^{(1)}]Y^{(1)}$ since $W^{(1)} = \emptyset$. In subsequent steps, the deflated algorithm is used instead of the standard CG using the set $W \equiv W^{(s)}$. The matrices $AW^{(s)}$ and $(W^{(s)})^T AW^{(s)}$ are computed using (5.9). We proceed as before except that $W^{(s+1)}$ is now defined by $W^{(s+1)} = [W^{(s)}, P_l^{(s)}]Y^{(s)}$. The matrices $G^{(s)}$ and $F^{(s)}$ to compute the eigenvectors $y_i^{(s)}$ are computed according to the formulas (5.7)–(5.9).

A preconditioned version of the method described above can be readily obtained by considering the split-preconditioned systems

$$L^{-1}AL^{-T}y^{(s)} = L^{-1}b^{(s)}, \qquad x^{(s)} = L^{-T}y^{(s)}, \qquad s = 1, 2, \ldots, \nu.$$

As in the case of preconditioned Deflated-CG, we apply the method to systems $L^{-1}AL^{-T}y^{(s)} = L^{-1}b^{(s)}$ for the y-variable and then redefine the original variables according to (3.13). Everything in (5.3) remains the same except $G^{(s)}$ which now becomes

$$G^{(s)} = \begin{bmatrix} \left(AW^{(s)}\right)^T M^{-1} AW^{(s)} & \left(W^{(s)}\right)^T AW^{(s)} \tilde{\Delta}_{l+1}^{(s)} \tilde{L}_l^{(s)} \\ \left(\tilde{\Delta}_{l+1}^{(s)} \tilde{L}_l^{(s)}\right)^T \left(W^{(s)}\right)^T AW^{(s)} & \tilde{G}^{(s)} \end{bmatrix}, \tag{5.10}$$

where $M = LL^T$.

Also the computation of $AP_l^{(s)}$ in (5.6) now becomes

$$M^{-1}AP_l^{(s)} = M^{-1}R_{l+1}^{(s)}\tilde{L}_l^{(s)} = \left(W^{(s)}\tilde{\Delta}_{l+1}^{(s)} + P_{l+1}^{(s)}\tilde{U}_l^{(s)}\right)\tilde{L}_l^{(s)}. \tag{5.11}$$

Putting these relations together we obtain the following algorithm.

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

ALGORITHM 5.3. Deflated PCG for multiple right-hand sides.

1. *Select l and k with $l \geq k$.*
2. *Solve the first system of* (5.1) *with the standard PCG.*
3. *Compute $G^{(1)}$ and $F^{(1)}$ using* (5.10) *and* (5.8). *Set $W^{(1)} = \emptyset$.*
4. *For $s = 2, 3, \ldots, \nu$, Do:*
5. *Solve* (5.3) *for k eigenvectors $y_1^{(s-1)}, y_2^{(s-1)}, \ldots, y_k^{(s-1)}$. Set $W^{(s)} = \left[W^{(s-1)}, P_l^{(s-1)}\right] Y^{(s-1)}$.*
6. *Choose $l \geq 0$.*
7. *Solve the sth system of* (5.1) *by preconditioned Deflated-CG with $W = W^{(s)}$. Compute $G^{(s)}$ and $F^{(s)}$ according to* (5.10), (5.7), (5.8), *and* (5.11).
8. *EndDo*

In addition to the memory required by the preconditioned Deflated-CG algorithm, $l+1$ vectors $\hat{\mu}_0^{(s)}$, $\hat{\mu}_1^{(s)}$, $\ldots$, $\hat{\mu}_l^{(s)}$ must be stored along with the three matrices $\tilde{L}_l^{(s)}$, $U_l^{(s)}$, and $\tilde{D}_l^{(s)}$.

6. Practical considerations. We now consider the memory and computational cost requirements of the deflated CG algorithm. We assume that $k \ll n$ so that we can neglect terms not containing n.

In Deflated-CG, we must store W and AW in addition to the usual vectors of CG. This means an additional storage of $2k$ vectors of length n. Deflated-CG requires computing Hr_j at each step. This can be done using common BLAS2 operations in $z_j = (AW)^T r_j$ and $r_j - W(W^T AW)^{-1} z_j$. The cost of these operations is $O(kn)$ so that the CPU overhead is not too high. There remains to consider the cost of computing W. In AugCG, which can be viewed as a particular case of Deflated-CG, W is the set of descent directions from the first system, so that the set W is A-orthogonal. The method has at least two advantages: Only the last column of W needs to be saved instead of all W. The projection H simplifies into only one orthogonalization (for only the last vector w_k). However, if s is greater than 2, then it is not easy to refine W. The only solution would be to store all consecutive sets of direction descents $W^{(s)}$ at a cost of a high memory requirement.

From a computational point of view, it is advantageous to have a set of vectors W that is A-orthogonal, in which case, the matrix $W^T AW$ becomes diagonal. However, this is not essential and the difference in cost involved is minimal.

Approximate eigenvectors are computed from the generalized eigenproblem (5.3). We need to store not only W and AW but also l vectors $P_l^{(s)}$, amounting to $2k + l$ vectors of length n. The computation of W using (5.5) and AW using (5.9) require mainly BLAS3 operations of complexity $O(n(l+1)k)$ and the computation of $W^T(AW)$ is a BLAS3 operation of complexity $O(nk^2)$. If k and l are kept small, the overhead is modest.

7. Numerical experiments. In this section, we present some examples to illustrate the numerical convergence behavior of Algorithm 5.3 and the analysis in section 5. All the experiments were performed in MATLAB with machine precision 10^{-16}. The stopping criterion for Examples 1–3 is that the relative residual $\|b - Ax_j\|_2/\|b\|_2$ be less than 10^{-7}. All the figures except Figure 7.4 plot the true relative residual versus the number of iterations taken.

Example 1. The purpose of this example is to test the analysis of section 5. We considered the matrix $A = Lapl(20, 20)$, a matrix of size $n = 400$ generated from a 5-point centered difference discretization of a Laplacian on a 22×22 mesh (20 points in

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

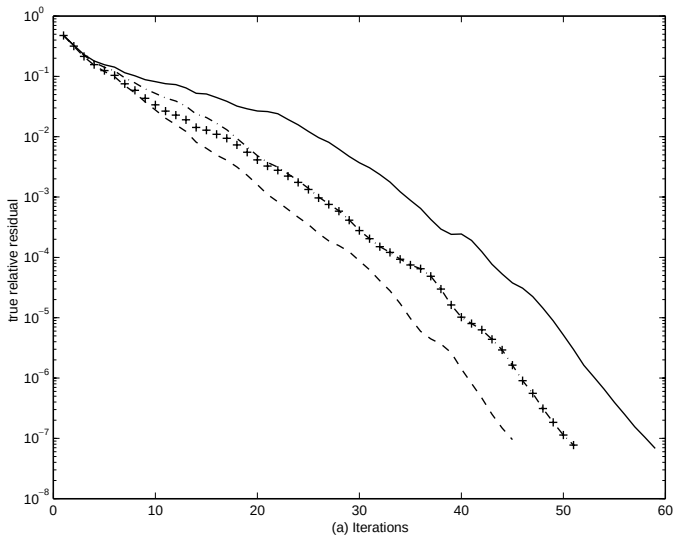

FIG. 7.1. *Example* 1.

each direction). The right-hand side b of (3.1) was chosen to be a random vector with independent and identically distributed (iid) entries from a normal distribution with mean 0 and variance 1 ($N(0,1)$). The largest eigenvalue of A is 7.9553 and the four smallest ones are $0.0447, 0.1112, 0.1112, 0.1777$. We first computed the exact eigenvectors w_1, w_2, w_3 associated with the three smallest eigenvalues $0.0447, 0.1112, 0.1112$, respectively, and then applied the standard CG with $x_0 = 0$, the Deflated-CG with $x_{-1} = 0$ in (3.12), and $W = [w_1], [w_1, w_2], [w_1, w_2, w_3]$, respectively, to the system (3.1). Their convergence behaviors are plotted in Figure 7.1 with solid, dashdot, plus, and dashed curves, respectively.

From Figure 7.1, we can see that the convergence behavior of Deflated-CG with $W = [w_1]$ is better than that of CG. Deflated-CG with $W = [w_1]$ solves a system with the condition number $\kappa = 7.9553/0.1112$. On the other hand, the convergence rates of Deflated-CG with $W = [w_1]$ and $W = [w_1, w_2]$, respectively, are almost the same since they are solving systems with the same condition numbers $\kappa = 7.9553/0.1112$. It can be understood that Deflated-CG with $W = [w_1, w_2, w_3]$ has the best behavior since the corresponding condition number $\kappa = 7.9553/1.777$ is the smallest.

Examples 2 and 3 demonstrate the efficiency of our algorithm when applied to (5.1). We always choose the initial guess $x_0 = 0$ in solving the first system and $x_{-1} = 0$ in (3.12) for the remaining systems solving. The number of right-hand sides of (5.1) is 10 and the $b^{(s)}$'s are independent random vectors with iid entries from $N(0,1)$. Although we have selected the right-hand sides independently, we believe same conclusions can be made when they are dependent. Also, we kept the data in the first $l = 20$ steps, and $k = 5$ approximate eigenvectors associated with smallest eigenvalues were calculated via the QZ algorithm in Matlab when we solved each system. We compared our algorithm with Algorithm AugCG [5] with $m = 30$ for the second system ($s = 2$). All the test matrices were from the Harwell–Boeing collection.[2]

[2]http://math.nist.gov/MatrixMarket/data/

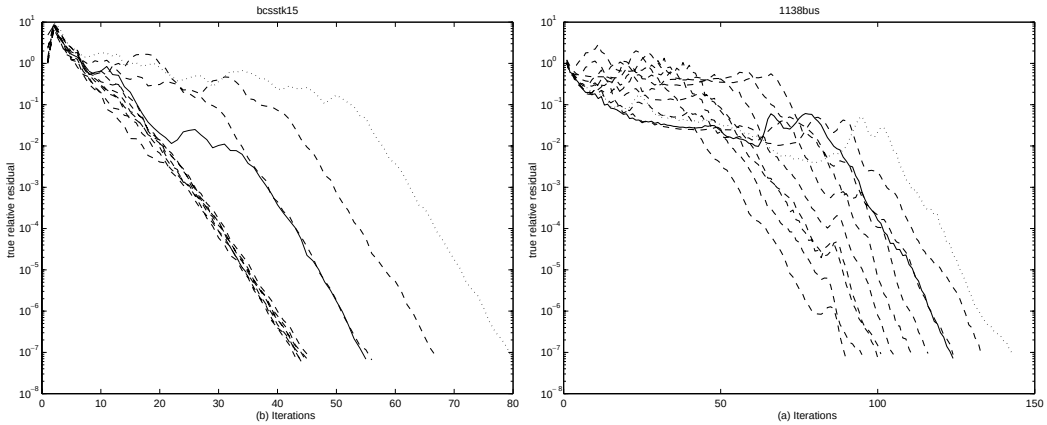

FIG. 7.2. (a) *Convergence curves for matrix BCSSTK*15 *of Example* 2. (b) *Convergence curves for matrix* 1138*bus of Example* 3.

The convergence behavior of system $s = 1$, corresponding to the standard preconditioned CG algorithm, is plotted with a dotted line. The convergence behaviors of systems $s = 2, \ldots, s = 10$ using Algorithm 5.3 are plotted with a dashed line. The convergence behavior of system $s = 2$ using Algorithm AugCG is plotted with a solid line.

Example 2. This example is the second matrix named BCSSTK15 from the BCSSTRUC2 group of the Harwell–Boeing collection. The order of the matrix is 3948. The system was preconditioned by incomplete Cholesky factorization $ic(1)$. Results are shown in Figure 7.2 (a).

Example 3. The matrix is the fourth one named 1138BUS from the PSADMIT group. The order of the matrix is 1138. The $ic(0)$ preconditioner was used and the results are shown in Figure 7.2 (b).

For both Examples 2 and 3, we observe that the number of iterations decreases significantly after the first few systems solving and quickly tends to its lower bound. This phenomenon is more remarkable when l and k are increased. It is because the approximate eigenvectors we chose quickly approach to their corresponding limits in the first few rounds of refinements. When these approximate eigenvectors have reached their limits, the number of iterations no longer decreases. The convergence speed of the approximate eigenvectors may depend on the distribution of eigenvalues. We computed several extreme eigenvalues of the matrices $L^{-1}AL^{-T}$ in both examples and found that the condition number λ_n/λ_1 and the number λ_{k+1}/λ_1 in Example 2 are both less than the corresponding ones in Example 3. To some degree, the number λ_{k+1}/λ_1 may reflect the conditions of the matrices $F^{(s)}$ and $G^{(s)}$ given by (5.4). This observation may help to explain why the approximate eigenvectors in Example 2 have faster convergences than those in Example 3.

In practice, we may omit line 5 or vary k and l when we find that the approximate eigenvectors are no longer improved when running Algorithm 5.3. However, the question of how to define a criterion to characterize the situation when eigenvectors no longer improve is still an open problem.

The algorithm does not always behave so well as demonstrated in the last two examples. The one below illustrates this situation.

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

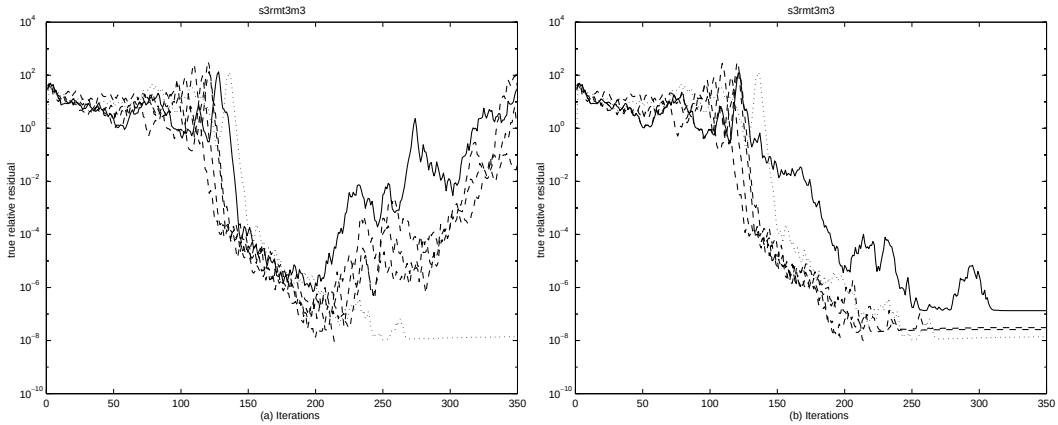

FIG. 7.3. (a) *Convergence curves for matrix S3RMT3M3 of Example* 4. (b) *Convergence curves with correction of* r_j *for matrix S3RMT3M3 of Example* 4.

Example 4. The test matrix is the one named S3RMT3M3 from the CYLSHELL group of `Independent Sets and Generators`.[3] In this experiment, we chose $l = k = 10$ for Algorithm 5.3 and $m = 30$ for Algorithm AugCG. Moreover, the incomplete Cholesky preconditioner $ic(2)$ was used and we set the stopping criterion $\|b - Ax_j\|_2/\|b\|_2 < 10^{-8}$ and the number of right-hand sides of (5.1) to be 5. Everything remained the same as in Examples 2 and 3 and the convergence behaviors were plotted in Figure 7.3 (a).

We observe from the experiment that systems 2, 3, and 5 solved by Deflated-CG and system 2 solved by AugCG do not converge. What is worse is that their convergence seems to be subject to instability. The situation is even worse for larger l and k. Theoretically, the residuals r_j in both Deflated-CG and AugCG are orthogonal to all the columns of W. In practice, however, this orthogonality is gradually lost as the algorithms progress. In fact, Figure 7.4 shows a plot of the function

$$othor(j) = \min\left(\frac{w_i^T r_j}{\|w_i\|_2\|r_j\|_2},\ i = 1, 2, \ldots, k\right)$$

for system 2 solved by AugCG and system 5 solved by Deflated-CG. Both curves of the function $othor(j)$ show that loss of orthogonality is so high that it ruins convergences.

One remedy to recover orthogonality is to add the reorthogonalization step

$$r_j := r_j - W\left(W^T W\right)^{-1} W^T r_j \tag{7.1}$$

right after r_j is computed in the algorithms. According to the analysis in section 6, the computational cost of the step is $O(kn)$. We ran Algorithm 5.3 and Algorithm AugCG again with this correction included and plotted the results in Figure 7.3 (b). This time, only systems 3 and 5 solved by Deflated-CG and system 2 solved by AugCG do not converge and all of them have decreasing convergence curves. By comparison, we also plotted the function $othor(j)$ after correcting r_j with (7.1) in Figure 7.4. Note that the curves with correction are much lower than those without correction.

The derivation of Deflated-CG is basically parallel to that of CG. In Deflated-CG, the standard Lanczos procedure is applied to the auxiliary matrix B in (2.3) and the

[3] http://math.nist.gov/MatrixMarket/data/

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

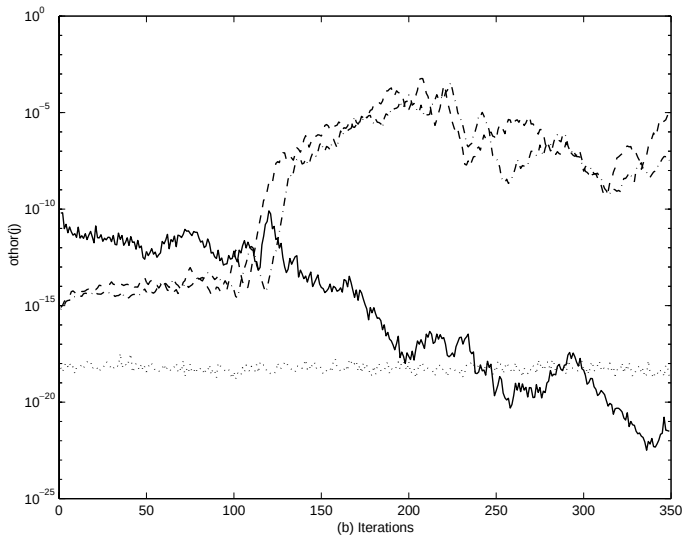

FIG. 7.4. *Orthogonality of r_j against W of Example* 4. *Dashed: system* 2 *solved by AugCG; dashdot: system* 5 *solved by Deflated-CG; solid: system* 2 *solved by AugCG with correction; dotted: system* 5 *solved by Deflated-CG with correction.*

matrix T_j in (2.4) is splitted into LDL^T decomposition. Unlike the case of CG, both B and T_j are not necessarily positive definite. As a result, we cannot expect that Deflated-CG will have the same robust behavior as CG does.

8. Conclusion. An algorithm was presented which incorporates deflation to the conjugate gradient algorithm with arbitrary systems of vectors. The method can be used for solving linear systems with multiple and dependent right-hand sides. The main advantage of this approach is that the size k of the subspace to be kept can be kept small without loss of efficiency relative to methods which require saving whole previous Krylov subspaces. Another advantage is that it is easy to refine the set W as each new system is solved. Theoretical results as well as experimentation confirm that convergence which results from the deflation improves substantially as the number of systems increases. As is expected, as soon as the set W of approximate eigenvectors is computed accurately, there is no further improvement for each new system to be solved unless the dimension of W is increases.

Acknowledgments. The authors wish to thank the referees and Dr. David Day for their valuable comments and discussions on this paper. Also, the first two authors would like to acknowledge the support of the Minnesota Supercomputer Institute which provided the computer facilities and an excellent environment to conduct this research.

REFERENCES

[1] S. F. ASHBY, T. A. MANTEUFFEL, AND P. E. SAYLOR, *A taxonomy for conjugate gradient methods*, SIAM J. Numer. Anal., 27 (1990), pp. 1542–1568.

[2] A. CHAPMAN AND Y. SAAD, *Deflated and augmented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43–66.

[3] M. O. BRISTEAU AND J. ERHEL, *Augmented conjugate gradient. Application in an iterative process for the solution of scattering problems,* Numer. Algorithms, 18 (1998), pp. 71–90.

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

[4] J. Erhel, K. Burrage, and B. Pohl, *Restarted GMRES preconditioned by deflation*, J. Comput. Appl. Math., 69 (1996), pp. 303–318.

[5] J. Erhel and F. Guyomarc'h, *An Augmented Subspace Conjugate Gradient*, Research report RR-3278 INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, 78150 Le Chesnay, France, 1997.

[6] W. D. Joubert and T. A. Manteuffel, *Iterative Methods for Nonsymmetric Linear Systems*, Academic Press, New York, 1990, pp. 149–171.

[7] S. A. Kharchenko and A. Yu. Yeremin, *Eigenvalue translation based preconditioners for the GMRES(k) method*, Numer. Linear Algebra Appl., 2 (1995), pp. 51–70.

[8] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[9] R. A. Nicolaides, *Deflation of Conjugate Gradients with Applications to Boundary Value Problems*, SIAM J. Numer. Anal., 24 (1987), pp. 355–365.

[10] B. Parlett, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 323–346.

[11] Y. Saad, *On the Lanczos method for solving symmetric linear systems with several right-hand sides*, Math. Comp., 178 (1987), pp. 651–662.

[12] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.

[13] Y. Saad, *Analysis of augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 435–449.

[14] E. De Sturler, *Truncation Strategies for Optimal Krylov Subspace Methods*, Technical Report TR-96-38, Swiss Center for Scientific Computing, Swiss Federal Institute of Technology, Zurich, Switzerland, 1996.

[15] E. De Sturler, *Inner-outer methods with deflation for linear systems with multiple right-hand sides*, Presentation in Householder Symposium XIII, 1996.

[16] H. van der Vorst, *An iterative method for solving $f(A)x = b$ using Krylov subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.

Downloaded 12/30/12 to 150.135.135.70. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php