Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 457089, 7 pages
http://dx.doi.org/10.1155/2013/457089

Research Article

A Polynomial Preconditioned Global CMRH Method for Linear Systems with Multiple Right-Hand Sides

Ke Zhang and Chuanqing Gu

Department of Mathematics, Shanghai University, Shanghai 200444, China

Correspondence should be addressed to Chuanqing Gu; cqgu@shu.edu.cn

Received 14 March 2013; Revised 23 June 2013; Accepted 26 September 2013

Academic Editor: Jinyun Yuan

The restarted global CMRH method (Gl-CMRH(m)) (Heyouni, 2001) is an attractive method for linear systems with multiple right-hand sides. However, Gl-CMRH(m) may converge slowly or even stagnate due to a limited Krylov subspace. To ameliorate this drawback, a polynomial preconditioned variant of Gl-CMRH(m) is presented. We give a theoretical result for the square case that assures that the number of restarts can be reduced with increasing values of the polynomial degree. Numerical experiments from real applications are used to validate the effectiveness of the proposed method.

1. Introduction

We consider the linear systems of the form

$$AX = B, \tag{1}$$

where A is a nonsingular and sparse matrix of order n, $B \in \mathbb{R}^{n\times s}$ with usually $s \leq n$. Such a situation arises from, for instance, wave-scattering problems, image restorations, recursive least squares computations, numerical methods for integral equations, and structural mechanics problems; see [1, 2] and references therein.

Numerical solvers for (1) can be roughly divided into two categories. The first is the *direct method*; for instance, the LU factorization is competent since A is only factorized once to recast (1) into a triangular system which is easy to solve. However, if the coefficient matrix A is large and sparse or sometimes not readily available, then *iterative solvers* may become the only choice and possibly fall into the following three classes.

The first class is the *block method*. For symmetric problems, the first block methods are due to O'Leary, including the block conjugate gradient method and block biconjugate gradient method [3]. For nonsymmetric cases, the block generalized minimal residual method [4–6], the block BiCGSTAB method [7], the block quasiminimal residual method [8], the block least squares method [9], the block Lanczos method [10], and the block IDR(s) method [11] have been proposed recently. In general, the block solvers are more suitable for dense systems with precondition.

The second class is the *seed method*. The main idea of this kind of methods is briefed below. We first select a single system (seed system) and develop the corresponding Krylov subspace. Then we project all the residuals of the other systems onto the same Krylov subspace to find new approximations as initial guess; see [2, 12, 13] for details.

The last class is the *global method*. To our knowledge, the term *global* is at least due to Saad [14, Chapter 10] and has been further populated by Jbilou et al. [15] with the global FOM and global GMRES methods for matrix equations. Following the work [15], many other global methods have been developed, including, to name just a few of them, the global BiCG and global BiCGSTAB methods [16, 17], the global Hessenberg and global CMRH (changing minimal residual method based on the Hessenberg process) methods [18] and their weighted variants [19], the skew-symmetric methods [20], and the global SCD method [21]. Generally, the global methods are more appropriate for large and sparse systems.

It is well known that the performance of the above Krylov subspace methods can be reinforced with a suitable preconditioner [14] or through effective matrix splitting techniques [22, 23]. In this paper, we are interested in preconditioning the global methods. Specifically, we aim at

improving the convergence behavior of the restarted global CMRH method (Gl-CMRH(m)) [18], which is originally proposed to reduce the increasing storage requirement in its full version. However, because of the use of a small subspace (say $m \ll n$), Gl-CMRH(m) is likely to slow down or even stalls out. Heyouni and Essai give a weighted version of Gl-CMRH(m) (WGl-CMRH(m)) [19] to alleviate such disadvantage. Instead, we propose a different approach, that is, by polynomial preconditioner to improve Gl-CMRH(m) in this paper.

The remainder of this work is organized as follows. In Section 2, we first recall some notations and properties of the global method, and then we sketch the Gl-CMRH(m) method. In Section 3, we construct the polynomial preconditioner tailored to Gl-CMRH(m) by exploiting the relation between the Krylov matrix and the global basis. For square right-hand side matrices, we also give a theoretical result that justifies the use of the proposed polynomial preconditioner. In Section 4, several numerical examples are employed to substantiate the effectiveness of the proposed method. Some concluding remarks and potential future work are briefed in the last section.

2. Notations and the Global CMRH Method

In this section, we first give some notations and properties used in the *global* methods, which will henceforth be adopted extensively in deriving the main results. Then we present a brief introduction of the Gl-CMRH(m) method [18]. More details about the global methods can be found, for instance, in [15, 18, 19].

2.1. Notations and Properties. Throughout this paper, the following notations will be used. The norms $\|\cdot\|_2$ and $\|\cdot\|_F$ represent the vector 2-norm and matrix F-norm, respectively. Let $\mathbb{M}$ be the set of $n \times s$ rectangular matrices. If $X \in \mathbb{M}$, then X^T stands for its transpose. For a square matrix A, A^{-1} indicates the inverse of A if existed. Unless otherwise stated, subscripts denote the corresponding iteration step; for example, X_k denotes the kth iterate of the matrix (vector) X. Moreover, the (i, j) entries of matrices Y and X_k are denoted by $(Y)_{i,j}$ and $(X_k)_{i,j}$, respectively. If a column or a row of a matrix is invoked, then we denote it in a dot format; that is, $(X_k)_{\cdot,j}$ and $(X_k)_{i,\cdot}$ mean correspondingly the jth column and the ith row of X_k. Besides, $(X_k)_{i:j,s:t}$ extracts the submatrix from i to j rows and from s to t columns of X_k.

Next we present some notations and basic properties used in the global methods [15]. Given the $n \times ms$ block matrix $\mathscr{V}_m = [V_1, V_2, \ldots, V_m]$, where $V_i \in \mathbb{M}, i = 1, 2, \ldots, m$, then we define the matrix product $*$ as

$$\mathscr{V}_m * f = \sum_{i=1}^{m} (f)_i V_i, \tag{2}$$

where $f \in \mathbb{R}^m$. For any matrix $H \in \mathbb{R}^{m\times m}$, we define analogously the $*$ product by

$$\mathscr{V}_m * H = \left[\mathscr{V}_m * (H)_{\cdot,1}, \mathscr{V}_m * (H)_{\cdot,2}, \ldots, \mathscr{V}_m * (H)_{\cdot,m}\right]. \tag{3}$$

It can be verified that such matrix product satisfies the following properties:

$$\begin{aligned} \mathscr{V}_m * (f + g) &= (\mathscr{V}_m * f) + (\mathscr{V}_m * g), \\ (\mathscr{V}_m * H) * f &= \mathscr{V}_m * (Hf), \end{aligned} \tag{4}$$

where $f, g \in \mathbb{R}^m$.

2.2. The Global CMRH Method. The Gl-CMRH method [18] is an efficient extension of the CMRH method [24] for solving (1). It is based on the global Hessenberg process [18]. As for the numerical performance, Gl-CMRH is in general competitive with the classic global GMRES method (Gl-GMRES) [15].

Now we give a brief sketch of Gl-CMRH. Let $X_0 \in \mathbb{M}$ be the initial guess of (1) with the associated residual matrix $R_0 = B - AX_0$. The mth iteration X_m is searched in the affine subspace $X_0 + \mathscr{K}_m(A, R_0)$; that is, $X_m - X_0 = W_m \in \mathscr{K}_m(A, R_0)$, where W_m is the mth correction matrix. The matrix Krylov subspace is defined as $\mathscr{K}_m(A, R_0) = \operatorname{span}\{R_0, AR_0, \ldots, A^{m-1}R_0\}$. Using the basis $\mathscr{V}_m$ of $\mathscr{K}_m(A, R_0)$ given by the global Hessenberg process [18], we get

$$X_m = X_0 + \mathscr{V}_m * y_m, \tag{5}$$

where $y_m \in \mathbb{R}^m$. The global Hessenberg process in [18] also yields

$$\begin{aligned} A\mathscr{V}_m &= \mathscr{V}_{m+1} * \overline{H}_m \\ &= \mathscr{V}_m * H_m \\ &\quad + \left(\overline{H}_m\right)_{m+1,m} [\mathbf{0}, \mathbf{0}, \ldots, \mathbf{0}, V_{m+1}], \end{aligned} \tag{6}$$

where $\overline{H}_m$ is an $(m + 1) \times m$ upper Hessenberg matrix, H_m is obtained by deleting the last row of $\overline{H}_m$, and $\mathbf{0}$ is the zero matrix in $\mathbb{M}$. Thus it follows immediately that

$$R_m = R_0 - A\mathscr{V}_m * y_m = \mathscr{V}_{m+1} * \left(\beta e_1^{(m+1)} - \overline{H}_m y_m\right), \tag{7}$$

where $\beta = \max_{1\le i\le n, 1\le j\le s}\{|(R_0)_{i,j}|\}$ and $e_1^{(m+1)} = [1, 0, \ldots, 0]^T \in \mathbb{R}^{m+1}$. To obtain the vector y_m, a restriction is imposed on the Gl-CMRH method; that is,

$$\|R_m\|_F = \min_{W\in\mathscr{K}_m(A,R_0)} \|R_0 - AW\|_F. \tag{8}$$

Relations (7) and (8) yield

$$y_m = \arg\min_{y\in\mathbb{R}^m} \left\|\mathscr{V}_{m+1} * \left(\beta e_1^{(m+1)} - \overline{H}_m y\right)\right\|_F. \tag{9}$$

Instead of solving (9), which requires $\mathcal{O}(nm^2)$ operations and $\mathcal{O}(nm)$ storage, we solve a smaller problem

$$\min_{y\in\mathbb{R}^m} \left\|\beta e_1^{(m+1)} - \overline{H}_m y\right\|_2, \tag{10}$$

which leads to $y_m = \beta(\overline{H}_m^T \overline{H}_m)^{-1} \overline{H}_m^T e_1^{(m+1)}$ by assuming that $\overline{H}_m$ is of full rank. From (5) and (10), the mth iterate X_m can be updated by

$$X_m = X_0 + \beta \mathscr{V}_m * \left(\left(\overline{H}_m^T \overline{H}_m \right)^{-1} \overline{H}_m^T e_1^{(m+1)} \right). \tag{11}$$

As in the Gl-GMRES method [15], a restarting strategy is used to address the problem that the computational and storage requirements increase with iterations. Algorithm 1 gives a framework of the restarted version of Gl-CMRH (Gl-CMRH(m)). We refer to [18, 19] for elaborate explanation for the Gl-CMRH method.

3. A New Polynomial Preconditioned Gl-CMRH(m) Method

In many cases, the accuracy of Gl-CMRH(m) is sufficient. Due to the limited dimension of the matrix Krylov subspace $\mathscr{K}_m$, however, Gl-CMRH(m) may suffer from slow convergence or even stagnation in practice, just like in GMRES(m) [25] and Gl-GMRES(m) [15]. To remedy this drawback, some accelerating techniques are demanded, for instance, a weighting strategy exploited in [19]. Besides, a polynomial preconditioner can also be adapted to improve the convergence [26]. In this section we focus on constructing an efficient polynomial preconditioner pertinent to the Gl-CMRH(m) method.

The essence of the polynomial preconditioned method is to devise a polynomial $Q(A) \approx A^{-1}$ such that an easier system

$$Q(A) AX = Q(A) B \tag{12}$$

is solved instead of solving the original system (1). In what follows, we obtain a polynomial preconditioner $Q(x)$ by extracting some useful information from Gl-CMRH(m).

Now suppose that the block Krylov matrix K_k is of the form

$$K_k = \left[R_0, AR_0, \ldots, A^{k-1} R_0 \right], \tag{13}$$

where R_0 is the initial residual matrix. By comparing the last s columns of the second equation in (6), we have

$$AV_k = \mathscr{V}_k * (H_k)_{\cdot,k} + \left(\overline{H}_k \right)_{k+1,k} V_{k+1}. \tag{14}$$

The equality (14) can be rearranged as

$$V_{k+1} = \left(\left(\overline{H}_k \right)_{k+1,k} \right)^{-1} \left(AV_k - \mathscr{V}_k * (H_k)_{\cdot,k} \right). \tag{15}$$

Let us consider the relationship between K_k and the basis $\mathscr{V}_k$. Since $\mathscr{V}_k$ and K_k span the same space, it follows that

$$\mathscr{V}_k = K_k * U_k, \tag{16}$$

where U_k is an upper triangular matrix. The relation (16), however, does not shed too much light because how to compute U_k still remains unclear. Fortunately, an explicit recurrence for U_k can be derived in terms of U_{k-1} and H_{k-1}. By combining (16) and (4), we get

$$\begin{aligned} \mathscr{V}_k * (H_k)_{\cdot,k} &= K_k * \left(U_k (H_k)_{\cdot,k} \right) \\ &= \left[K_k, A^k R_0 \right] * \begin{pmatrix} U_k (H_k)_{\cdot,k} \\ 0 \end{pmatrix}. \end{aligned} \tag{17}$$

Since $V_k = (\mathscr{V}_k)_{\cdot,(k-1)s+1:ks} = K_k * (U_k)_{\cdot,k}$, then

$$AV_k = \left[AR_0, A^2 R_0, \ldots, A^k R_0 \right] * (U_k)_{\cdot,k} = K_{k+1} * \begin{pmatrix} 0 \\ (U_k)_{\cdot,k} \end{pmatrix}. \tag{18}$$

Substituting (17) and (18) into (15) gives rise to

$$V_{k+1} = \left(\left(\overline{H}_k \right)_{k+1,k} \right)^{-1} K_{k+1} * \left(\begin{pmatrix} 0 \\ (U_k)_{\cdot,k} \end{pmatrix} - \begin{pmatrix} U_k (H_k)_{\cdot,k} \\ 0 \end{pmatrix} \right). \tag{19}$$

Besides, the relation (16) gives $V_{k+1} = K_{k+1} * (U_{k+1})_{\cdot,k+1}$. By combining it with (19), we obtain a recurrence for the $(k+1)$st column of U_{k+1}; that is,

$$(U_{k+1})_{\cdot,k+1} = \left(\left(\overline{H}_k \right)_{k+1,k} \right)^{-1} \left(\begin{pmatrix} 0 \\ (U_k)_{\cdot,k} \end{pmatrix} - \begin{pmatrix} U_k (H_k)_{\cdot,k} \\ 0 \end{pmatrix} \right). \tag{20}$$

Therefore, U_k in (16) can be updated recursively by (20). Recall that in (5) X_k is updated on the basis $\mathscr{V}_k$. Here we will show another way to update X_k which is based on the block Krylov matrix K_k. It follows from (5), (16), and (4) that

$$\begin{aligned} X_k &= X_0 + \left(K_k * U_k \right) * y_k = X_0 + K_k * \left(U_k y_k \right) \\ &= X_0 + \sum_{i=0}^{k-1} \alpha_i A^i R_0, \end{aligned} \tag{21}$$

where $U_k y_k = [\alpha_0, \alpha_1, \ldots, \alpha_{k-1}]^T$ and y_k is solved from (10). Denote by Q_{k-1} a polynomial in A of degree $k-1$; that is, $Q_{k-1}(A) = \sum_{i=0}^{k-1} \alpha_i A^i$. Hence (21) can be recast as

$$X_k = X_0 + Q_{k-1}(A) R_0. \tag{22}$$

The matrix polynomial $Q_{k-1}(A)$ in (22) can be regarded as the approximation to A^{-1} in some sense. This is justified for the case $n = s$ in (1) by the following result.

Theorem 1. *Let $Q_{k-1}(A)$, X_0 and $R_0 = B - AX_0$ be the square matrices defined in* (22)*, and lot X_* be the true solution of* (1)*. Suppose that $X_* - X_0$ is nonsingular. Then one has*

$$\left\| I - Q_{k-1}(A) A \right\|_F \le \left\| E_k \right\|_F \left\| E_0^{-1} \right\|_F, \tag{23}$$

where $E_k := X_* - X_k$ and $E_0 := X_* - X_0$.

Proof. The inequality (23) follows immediately from an arrangement of (22). □

(1) Choose an initial guess X_0, the restarting frequency m and the tolerance tol. Compute $R_0 = B - AX_0$. Determine i_0, j_0 such that $\left|(R_0)_{i_0,j_0}\right| = \max_{1\le i\le n, 1\le j\le s}\{(R_0)_{i_0,j_0}\}$. Set $\beta = \left|(R_0)_{i_0,j_0}\right|$, $V_1 = R_0/\beta$, $p_{1,1} = i_0$ and $p_{1,2} = j_0$.
(2) Construct the matrix basis $\mathscr{V}_m = [V_1, V_2, \ldots, V_m]$ and $\overline{H}_m$ by the global Hessenberg process [18].
(3) Solve y_m by (10) and update X_m by (5).
(4) Compute $R_m = B - AX_m$. If $\|R_m\|_F/\|R_0\|_F \le$ tol, then stop; otherwise set $X_0 = X_m$, $R_0 = R_m$.
Choose i_0, j_0 such that $\left|(R_0)_{i_0,j_0}\right| = \max_{1\le i\le n, 1\le j\le s}\{(R_0)_{i_0,j_0}\}$.
Set $\beta = \left|(R_0)_{i_0,j_0}\right|$, $V_1 = R_0/\beta$, $p_{1,1} = i_0$ and $p_{1,2} = j_0$. Go to Step 2.

Algorithm 1: The Gl-CMRH(m) method for $AX = B$ [18].

Remark 2. In (23), the term $\|E_k\|_F$ becomes smaller with growing k and hence the upper bound diminishes correspondingly, which in turn implies that $Q_{k-1}(A)$ approximates A^{-1} asymptotically. This justifies the use of the polynomial preconditioner. In general, (23) assures that the number of restarts will be reduced correspondingly with increasing k. Yet this does not necessarily mean that the CPU time will be reduced simultaneously since the time saved from the reduction of restarts may be offset by the extra time spent in constructing the polynomial. In practice, we are often more concerned with the CPU time than the restarting number. Therefore, we restrict ourselves to small values of k. For $s < n$, an inequality similar to (23) is generally unavailable. Nevertheless, numerical examples seem to demonstrate that the asymptotical property of (23) is also shared by the case $s < n$; see Example 3 in Section 4 for more discussions.

By putting all together, we propose the new polynomial preconditioned global CMRH method (PGl-CMRH(m, deg)) that is shown in Algorithm 2.

4. Numerical Examples

In this section, we present some numerical experiments which are coded with MATLAB 7.8.0. For fair comparisons, some other global solvers mentioned earlier like Gl-CMRH(m) [18], WGl-CMRH(m) [19], and Gl-GMRES(m) [15] have also been implemented. From now on we drop the parameters m and deg in brackets without ambiguity. In all examples, we assume that $X_0 = \mathbf{0}$. The terminating criterion for the kth iteration is tol $= \|R_k\|_F/\|R_0\|_F \le 10^{-10}$. Though other alternatives are possible, we use $D = \sqrt{ns}|(R_0)_{i,j}|/\|R_0\|_F$ as the the weighting matrix for WGl-CMRH, which is also preferred in [19]. The coefficient matrices A in the first two examples are derived from the discretizations of the Poisson's equation and convection-diffusion problems which occur frequently in applied science and engineering. The coefficient matrices A in the third example are quoted from the Matrix Market [27].

Example 1. We consider the linear systems of (1) in which its coefficient matrix A is obtained from the discretization of

$$\mathscr{L}u = u_{xx} + u_{yy} \tag{24}$$

Table 1: Number of restarts and CPU time (in brackets) for Example 1.

n	Gl-GMRES	Gl-CMRH	WGl-CMRH	PGl-CMRH
10,000	121 (18.0)	85 (7.4)	89 (8.4)	24 (4.1)
14,400	150 (34.6)	85 (11.8)	116 (17.3)	23 (6.0)
22,500	259 (133.4)	165 (42.8)	173 (53.5)	37 (17.0)
40,000	450 (585.8)	255 (173.0)	302 (235.2)	26 (32.3)
44,100	496 (699.9)	322 (253.3)	368 (313.6)	39 (45.0)

on the unit square $(0, 1) \times (0, 1)$ with $u = 0$ on the boundary. It can be discretized through the centered difference scheme at the gird points (x_i, y_i) with $x_i = ih, y_j = jh$, where the mesh size $h = 1/(N + 1)$ for $i, j = 0, \ldots, N + 1$. This yields a block tridiagonal matrix of size $n = N^2$. The right-hand side matrix B is chosen with entries uniformly distributed on $[0, 1]$; see [14, Chapter 2] for more details about (24). Related parameters are given by $s = 2$, $m = 20$ and deg $= 5$. The number of restarts and CPU time for matrices A of different sizes are given in Table 1. As observed from Table 1, PGl-CMRH improves the original CMRH method by time ratios from 17.8% to 55.4%. Compared with WGl-CMRH, PGl-CMRH requires less number of restarts and CPU time to achieve the required accuracy. Note that WGl-CMRH does not speed up the convergence of Gl-CMRH. This indicates that a different weighting matrix should be used. To find the optimal weighting matrix, however, remains an open problem [19].

Example 2. Consider the linear systems of (1) where its coefficient matrix A is obtained from the discretization of the three-dimensional convection-diffusion problem

$$\mathscr{T}u = -\left(u_{xx} + u_{yy} + u_{zz}\right) + q\left(u_x + u_y + u_z\right) \tag{25}$$

on the unit cube $\Omega = [0, 1] \times [0, 1] \times [0, 1]$. Here q is a constant coefficient and (25) subjects to Dirichlet-type boundary conditions. This equation can be discretized by applying seven-point finite difference discretizations. For instance, we use the centered difference to the diffusive terms and the first-order upwind approximations to the convective terms. This approach yields a coefficient matrix A of size $n = N^3$, where the equidistant mesh size $h = 1/(N + 1)$ is used, and the natural lexicographic ordering is adopted to

(1) **Input**: m, deg and X_0.
(2) **Phase I**: Compute $R_0 = B - AX_0$. Determine i_0, j_0 such that $(R_0)_{i_0,j_0} = \max_{1\le i\le n, 1\le j\le s}\{|(R_0)_{i,j}|\}$.
Set $\beta = (R_0)_{i_0,j_0}$, $V_1 = R_0/\beta$, $p_{1,1} = i_0$ and $p_{1,2} = j_0$. Let $U_{1,1} = (R_0)_{i_0,j_0}$ (the upper triangular matrix defined in (16)).
(3) **for** $k = 1 : \deg$
(4) $M = AV_k$
(5) **for** $j = 1 : k$
(6) $(\overline{H})_{j,k} = (M)_{p_{j,1},p_{j,2}}$
(7) $M = M - (\overline{H})_{j,k}V_j$
(8) **end for**
(9) Determine i_0, j_0 such that $(M)_{i_0,j_0} = \max_{1\le i\le n, 1\le j\le s}\{|(M)_{i,j}|\}$.
(10) Set $(\overline{H})_{k+1,k} = (M)_{i_0,j_0}$, $V_{k+1} = M/(\overline{H})_{k+1,k}$, $p_{k+1,1} = i_0$ and $p_{k+1,2} = j_0$.
(11) Compute $(U_{k+1})_{:,k+1}$ from (20).
(12) **end for**
(13) $X_{\deg} = X_0 + \mathscr{V}_{\deg} * y_{\deg}$, where $y_{\deg} = \arg\min_{y\in\mathbb{R}^{\deg}} \left\|(R_0)_{i_0,j_0}e_1^{(\deg+1)} - \overline{H}_{\deg}y\right\|_2$.
(14) Construct the polynomial preconditioner $Q_{\deg-1}(A)$ with its coefficients decided by entries of $U_{\deg}y_{\deg}$.
(15) **Phase II**: Solve $Q_{\deg-1}(A)AX = Q_{\deg-1}(A)B$ using Algorithm 1.

Algorithm 2: PGl-CMRH(m, deg) method for $AX = B$.

Table 2: Number of restarts and CPU time for Example 2; $q = 0.1$ (top) and $q = 1$ (bottom).

n	Gl-GMRES	Gl-CMRH	WGl-CMRH	PGl-CMRH
8,000	14 (1.2)	11 (0.6)	13 (0.7)	2 (0.3)
27,000	26 (8.4)	23 (5.0)	21 (4.9)	5 (2.3)
64,000	40 (40.7)	32 (21.2)	32 (22.8)	7 (9.5)
125,000	58 (147.2)	41 (53.3)	47 (74.6)	9 (23.2)
216,000	81 (298.8)	58 (122.5)	61 (160.7)	17 (74.4)
n	Gl-GMRES	Gl-CMRH	WGl-CMRH	PGl-CMRH
8,000	14 (1.2)	13 (0.6)	14 (0.7)	2 (0.3)
27,000	25 (7.4)	22 (4.6)	22 (4.8)	5 (2.4)
64,000	39 (37.0)	32 (20.6)	34 (23.3)	7 (9.9)
125,000	57 (114.8)	43 (53.6)	48 (70.8)	9 (23.3)
216,000	79 (313.1)	51 (114.7)	61 (181.0)	17 (76.2)

Table 3: Properties of testing matrices in Example 3.

Matrix	n	nnz	Discipline
add32	4960	19848	Electronic circuit design
cdde6	961	4681	Computational fluid dynamics
fs680.1	680	2184	Chemical kinetics
fidap001	216	4339	Finite element modeling
gre115	115	421	Simulation studies in computer systems
pde2961	2961	14585	Partial differential equations
rdb200	200	1120	Chemical engineering
rdb2048l	2048	12032	Chemical engineering
rdb3200l	3200	18880	Chemical engineering
sherman4	1104	3786	Oil reservoir modeling

the unknowns; we refer to [22, Section 4] for more details. The right-hand side matrix B is chosen with entries uniformly distributed on $[0, 1]$. Here, $s = 2$, $m = 15$, and deg $= 5$. The number of restarts and CPU time for $q = 0.1$ and $q = 1$ is given in Table 2. For this large problem, as expected, PGl-CMRH performs better than CMRH and other variants concerning CPU time.

Example 3. In practice, the degree of the polynomial preconditioner $Q_{\deg-1}$ has a great impact on the numerical performance of PGl-CMRH. Thus it deserves our attention to investigate how to choose the "optimal" degree (if existed) for generic matrices. Nevertheless, theoretical analysis to this end can be very hard. Instead, we show empirically how to choose a range of degrees for the polynomial $Q_{\deg-1}$ such that PGl-CMRH at least yields a modest performance. To this end, we use ten unsymmetrical testing matrices from [27] and illustrate how PGl-CMRH performs for each matrix with deg varying from 2 to 15; see Figure 1. Some properties of these testing matrices are listed in Table 3. The right-hand side matrix B is chosen with entries uniformly distributed on $[0, 1]$. Since we are only concerned with the value of deg that makes PGl-CMRH performs stably with the shortest CPU time, we have normalized values of CPU time by dividing the maximum value of CPU time for a certain curve. Take the matrix pde2961 for example. The longest time is 4.2 seconds (with deg $= 3$); then we divide all values of CPU time by 4.2 for pde2961 and plot the result in Figure 1. This approach facilitates our comparison since different curves become more clustered now. Some remarks can be made from Figure 1. First, the curves seem rather problem-dependent and are not necessarily nonincreasing with increasing values of deg; for instance, the curve of

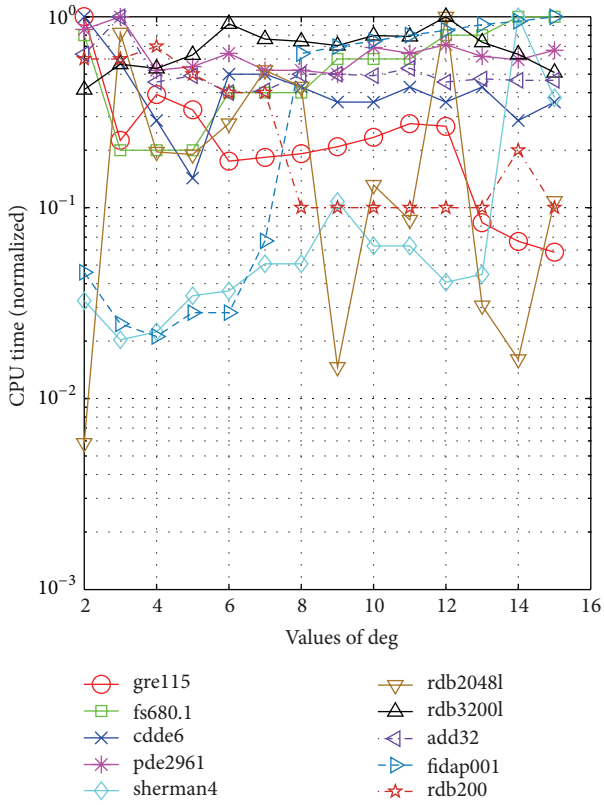

Figure 1: Normalized CPU time against deg (from 2 to 15) for ten testing matrices.

rdb2048l is rather irregular and hence unpredictable. However, this does not contradict Theorem 1 where it is stated that $Q_{\deg-1}(A)$ can approximate A^{-1} better with growing values of deg. In other words, Theorem 1 explains theoretically that the total number of restarts will be reduced with increasing values of deg. However, this does not apply to the change of CPU time. In fact, it is likely that PGl-CMRH with high degree preconditioner takes more CPU time in generating the polynomial preconditioner (even with less number of restarts) and hence uses more time to converge than that of its low degree counterpart. Second, most curves locate the corresponding shortest CPU time point with deg between 2 and 10. This can be the first reason for favoring small values of deg. Finally, more rounding errors can be introduced in developing high-degree polynomial preconditioners from the numerical point of view. This is the second reason for the approval of low-degree polynomial preconditioners. Therefore it is useful to test with deg from 2 to 10. Under extreme situations, however, higher degree may be demanded if a low-degree preconditioner fails to bring the required accuracy.

5. Conclusion

To remedy the slow convergence of the original Gl-CMRH(m) method, a new variant of Gl-CMRH(m) for linear systems with multiple right-hand sides is developed. The proposed method often yields better performance than its predecessor Gl-CMRH(m) and other global variants in terms of CPU time. We show experimentally that polynomial preconditioners with degree lower than 10 should be considered if no prior knowledge is known.

Acknowledgments

The authors would like to thank Professor Jinyun Yuan and the referees for their valuable remarks that improved this paper. The work is supported by the National Natural Science Foundation (11371243), the Key Disciplines of Shanghai Municipality (S30104), the Innovation Program of Shanghai Municipal Education Commission (13ZZ068), and the Anhui Provincial Natural Science Foundation (1308085QF117).

References

[1] T. F. Chan and M. K. Ng, "Galerkin projection methods for solving multiple linear systems," *SIAM Journal on Scientific Computing*, vol. 21, no. 3, pp. 836–850, 1999.

[2] T. F. Chan and W. L. Wan, "Analysis of projection methods for solving linear systems with multiple right-hand sides," *SIAM Journal on Scientific Computing*, vol. 18, no. 6, pp. 1698–1721, 1997.

[3] D. P. O'Leary, "The block conjugate gradient algorithm and related methods," *Linear Algebra and Its Applications*, vol. 29, pp. 293–322, 1980.

[4] B. Vital, *Etude de quelques méthodes de résolution de problµemes linéaires de grande taille sur multiprocesseurs [Ph.D. thesis]*, Universit de Rennes, Rennes, France, 1990.

[5] V. Simoncini and E. Gallopoulos, "An iterative method for nonsymmetric systems with multiple right-hand sides," *SIAM Journal on Scientific Computing*, vol. 16, no. 4, pp. 917–933, 1995.

[6] V. Simoncini and E. Gallopoulos, "Convergence properties of block GMRES and matrix polynomials," *Linear Algebra and Its Applications*, vol. 247, pp. 97–119, 1996.

[7] A. El Guennouni, K. Jbilou, and H. Sadok, "A block version of BICGSTAB for linear systems with multiple right-hand sides," *Electronic Transactions on Numerical Analysis*, vol. 16, pp. 129–142, 2003.

[8] R. W. Freund and M. Malhotra, "A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides," *Linear Algebra and Its Applications*, vol. 254, no. 1–3, pp. 119–157, 1997.

[9] S. Karimi and F. Toutounian, "The block least squares method for solving nonsymmetric linear systems with multiple right-hand sides," *Applied Mathematics and Computation*, vol. 177, no. 2, pp. 852–862, 2006.

[10] A. El Guennouni, K. Jbilou, and H. Sadok, "The block Lanczos method for linear systems with multiple right-hand sides," *Applied Numerical Mathematics*, vol. 51, no. 2-3, pp. 243–256, 2004.

[11] L. Du, T. Sogabe, B. Yu, Y. Yamamoto, and S.-L. Zhang, "A block *IDR*(s) method for nonsymmetric linear systems with multiple right-hand sides," *Journal of Computational and Applied Mathematics*, vol. 235, no. 14, pp. 4095–4106, 2011.

[12] C. F. Smith, A. F. Peterson, and R. Mittra, "Conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields," *IEEE Transactions on Antennas and Propagation*, vol. 37, no. 11, pp. 1490–1493, 1989.

[13] A. M. Abdel-Rehim, R. B. Morgan, and W. Wilcox, "Improved seed methods for symmetric positive definite linear equations with multiple right-hand sides," *Numerical Linear Algebra with Applications*, 2013.

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, Pa, USA, 2nd edition, 2003.

[15] K. Jbilou, A. Messaoudi, and H. Sadok, "Global FOM and GMRES algorithms for matrix equations," *Applied Numerical Mathematics*, vol. 31, no. 1, pp. 49–63, 1999.

[16] K. Jbilou and H. Sadok, "Global Lanczos-based methods with applications," Tech. Rep. LMA 42, Université du Littoral, Calais, France, 1997.

[17] K. Jbilou, H. Sadok, and A. Tinzefte, "Oblique projection methods for linear systems with multiple right-hand sides," *Electronic Transactions on Numerical Analysis*, vol. 20, pp. 119–138, 2005.

[18] M. Heyouni, "The global Hessenberg and CMRH methods for linear systems with multiple right-hand sides," *Numerical Algorithms*, vol. 26, no. 4, pp. 317–332, 2001.

[19] M. Heyouni and A. Essai, "Matrix Krylov subspace methods for linear systems with multiple right-hand sides," *Numerical Algorithms*, vol. 40, no. 2, pp. 137–156, 2005.

[20] C. Gu and H. Qian, "Skew-symmetric methods for nonsymmetric linear systems with multiple right-hand sides," *Journal of Computational and Applied Mathematics*, vol. 223, no. 2, pp. 567–577, 2009.

[21] C. Gu and Z. Yang, "Global SCD algorithm for real positive definite linear systems with multiple right-hand sides," *Applied Mathematics and Computation*, vol. 189, no. 1, pp. 59–67, 2007.

[22] Z.-Z. Bai, G. H. Golub, and M. K. Ng, "Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 24, no. 3, pp. 603–626, 2003.

[23] Z.-Z. Bai, G. H. Golub, L.-Z. Lu, and J.-F. Yin, "Block triangular and skew-Hermitian splitting methods for positive-definite linear systems," *SIAM Journal on Scientific Computing*, vol. 26, no. 3, pp. 844–863, 2005.

[24] H. Sadok, "CMRH: a new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm," *Numerical Algorithms*, vol. 20, no. 4, pp. 303–321, 1999.

[25] Y. Saad and M. H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[26] M. B. van Gijzen, "A polynomial preconditioner for the GMRES algorithm," *Journal of Computational and Applied Mathematics*, vol. 59, no. 1, pp. 91–107, 1995.

[27] Matrix Market, http://math.nist.gov/MatrixMarket/.