RC24698 (W0812-001) December 1, 2008
Mathematics

IBM Research Report

New Krylov-Subspace Solvers for Hermitian Positive Definite Matrices with Indefinite Preconditioners

Haim Avron[1], Anshul Gupta[2], Sivan Toledo[1]

[1]Tel-Aviv University

[2]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

NEW KRYLOV-SUBSPACE SOLVERS FOR HERMITIAN POSITIVE DEFINITE MATRICES WITH INDEFINITE PRECONDITIONERS

HAIM AVRON, ANSHUL GUPTA, AND SIVAN TOLEDO

Abstract. Incomplete LDL^* factorizations sometimes produce an indefinite preconditioner even when the input matrix is Hermitian positive definite. The two most popular iterative solvers for Hermitian systems, MINRES and CG, cannot use such preconditioners; they require a positive definite preconditioner. We present two new Krylov-subspace solvers, a variant of MINRES and a variant of CG, both of which can be preconditioned using any non-singular Hermitian matrix as long as the original system is positive definite. These algorithms allow the use of incomplete-factorization preconditioners for Hermitian positive-definite systems, even when the preconditioner is indefinite, and without resorting to a more expensive non-symmetric iterative Krylov-space solver.

1. Introduction

Algorithms that produce preconditioners for Hermitian positive-definite matrices sometimes generate indefinite preconditioners. Consider, for example, incomplete-Cholesky factorizations. Even if the original matrix is positive definite, dropping elements during the incomplete factorization can lead to a zero or negative diagonal element; the factorization breaks down. If we compute an incomplete LDL^* factorization instead, a negative diagonal does not break down the factorization (but a zero or tiny diagonal value does lead to instability). But a negative diagonal value in an $M = LDL^*$ factorization implies that the preconditioner M is indefinite. This is a problem, because symmetric Krylov-space solvers like CG [11] and MINRES [14] cannot use an indefinite preconditioner due to possible breakdown, either because of a divide by zero (CG) or due to taking the square root of a negative value (MINRES). Furthermore, the correctness proof of both algorithms rely on the existance of a Cholesky factor of the preconditioner [16].

There are two existing solutions to this problem (apart from computing another preconditioner); we propose a third. One existing solution is to use the computed preconditioner with an unsymmetric Krylov-space solver, like GMRES [17], QMR [6, 7] and BiCGTAB [20] (see [16] for more algorithms). However, using GMRES is expensive due to the long recurrence (expensive orthogonalization steps and a high memory requirement), and algorithms like QMR or BiCGSTAB do not minimize a norm of the residual or norm of the error as GMRES, CG, and MINRES do. In general it is not possible to get both optimality and a short recurrence with a non-symmetric method [5].

A second existing approach is to force positive definiteness by modifying the factorization process. Benzi's survey of these methods [3] notes that the various techniques tend to fall into two categories: simple and inexpensive fixes that result in low-quality preconditioners (in terms of rates of convergence), or sophisticated and expensive strategies that yield high-quality preconditioners. Another technique to force positive definiteness is to perturb the factorization after it is computed. When solving lest-squares problem using LSQR [15] each perturbation adds, at most, one or two iterations [2]. This is probably also true for solving symmetric positive definite matrices using CG.

Date: November 2008.

Algorithm 1 U-Conjugate Arnoldi Iteration

$b =$ arbitrary, $q_1 = b/\|b\|_U$
for $n = 1, 2, 3, \ldots$
  $v = Aq_n$
  for $j = 1$ to n
    $h_{jn} = q_j^* U v$
    $v = v - h_{jn} q_j$
  end for
  $h_{n+1,n} = \|v\|_U$ (if $h_{n+1,n} = 0$ then the algorithm fails).
  $q_{n+1} = v/h_{n+1,n}$

We propose a third way: new Krylov-subspace variants of CG and MINRES that guarantee convergence and which allow an indefinite preconditioner to be used.

The rest of the paper is organized as follows. Section 2 describes the U-conjugate Arnoldi Iteration, a tool we are going to use to develop the new algorithms. Section 3 presents the new variant of CG. Section 4 presents the new variant of MINRES. In Section 5 we show how the U-conjugate Arnoldi Iteration can be used to derive a couple of older algorithms. Extensive numerical experiments are reported in Section 6. We present our conclusions in Section 6.

2. The U-conjugate Arnoldi Iteration

The main tool that we use is a generalization of the classical Arnoldi iteration. The classical Arnoldi iteration forms, at step n, matrices Q_{n+1} and $\tilde{H}_n$ such that

$$AQ_n = Q_{n+1}\tilde{H}_n$$

where $\tilde{H}_n$ is upper Hessenberg and Q_{n+1} is unitary. Instead of requiring Q_n to be unitary we require it to be unitary relative to the U-norm, where U is an Hermitian positive definite matrix. That is, we replace the condition

$$Q_n^* Q_n = I_{n\times n}$$

with the condition

$$Q_n^* U Q_n = I_{n\times n}\,.$$

To do so, all we need to do is replace dot-products with U inner-products, and 2-norms with U-norms. See Algorithm 1 for the pseudo-code. It is easy to see that the classical Arnoldi iteration is the U-conjugate iteration with $U = I_{n\times n}$ (the identity matrix).

Like the classical Arnoldi iteration the U-conjugate Arnoldi iteration vectors span the the Krylov subspace. We omit the proof because it is identical to the proof that the classical Arnoldi iteration vectors span the Krylov subspace.

Theorem 2.1. *Let $q_1, \ldots, q_n$ be n vectors generated by a successful application of n iterations of Algorithm 1 on matrix A with initial vector b. We have*

$$\mathit{span}\,\{q_1, q_2, \ldots, q_n\} = \mathcal{K}_n(A, b)\,.$$

The following theorem summarizes a few useful properties of the values generated by the U-conjugate Arnoldi iteration.

Theorem 2.2. *Let $\{q_i\}$and $\{h_{ij}\}$ be the values generated by the successful application of n iterations of Algorithm 1 on matrix A with initial vector b. Denote*

$$Q_n = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}$$

$$\tilde{H}_n = \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ h_{21} & & \vdots \\ & \ddots & \vdots \\ & & h_{n+1} \end{bmatrix}$$

$$H_n = \left(\tilde{H}_n\right)_{1:n,1:n}$$

We have

(1) $AQ_n = Q_{n+1}\tilde{H}_n$
(2) $Q_n^* U Q_n = I_{n\times n}$
(3) $Q_n^* U A Q_n = H_n$

Proof. The first two properties follow directly from the algorithm. Multiply the equation in property 1 by $Q_n^* U$ to get

$$Q_n^* U A Q_n = Q_n^* U Q_{n+1} \tilde{H}_n \,.$$

It is easy to see that

$$Q_n^* U Q_{n+1} = \begin{bmatrix} I_{n\times n} & 0_{n\times 1} \end{bmatrix},$$

so we have $Q_n^* U A Q_n = H_n$. □

The U-conjugate Arnoldi has a major disadvantage: the amount of work required to perform the tth iteration and amount of memory space needed is $O(tn + \mathrm{nnz}(A))$. The classical Arnoldi reduces to a 3-term recurrence if A is Hermitian. The U-conjugate Arnoldi iteration reduces to a three term recurrence if $H_n = Q_n^* U A Q_n$ is Hermitian. This happens when UA is Hermitian. When this is the case, we call the resulting iteration *the U-Conjugate Lanczos Iteration* and use T_n instead of H_n.

3. Indefinitely Preconditioned Conjugate Gradients

If A is a Hermitian positive definite matrix we can use the U-conjugate Lanczos iteration to find an optimal A-norm approximate solution to $Ax = b$. We do so by applying the iteration on A, selecting $U = A$. After iteration n we have an A-conjugate basis to the Krylov subspace $\mathcal{K}_n(A, b)$. We can use the Conjugate Directions method to produce an optimal A-norm approximation. See §7 in [18].

This algorithm can be preconditioned quite easily. Suppose that we have formed an Hermitian preconditioner M. We can apply the A-conjugate Lanczos iteration to $M^{-1}A$ since $AM^{-1}A$ is Hermitian. Assuming we start our iteration with $M^{-1}b$, after the nth iteration we will find an n-dimensional A-conjugate basis to $\mathcal{K}(M^{-1}A, M^{-1}b)$. We can use that basis to find an optimal A-norm approximate solution $M^{-1}Ax = M^{-1}b$. We include the pseudo-code for the new algorithm in the Appendix (Algorithm 2). We refer to this algorithm as IP-CG from here on.

If both M and A are Hermitian positive definite then the classical CG produces the optimal A-norm approximate in $\mathcal{K}_n(M^{-1}A, M^{-1}b)$, that is, a minimizer of $\|x_n - x\|_A$ subject to $x_n \in \mathcal{K}_n(M^{-1}A, M^{-1}b)$. This minimizer is unique. This implies that under exact arithmetic, if the preconditioner is definite, both classical CG and IP-CG will produce the same vectors. Therefore, Algorithm 2 is indeed a different, and more robust, formulation of CG; it is an "Indefinitely Preconditioned Conjugate Gradients".

IP-CG's advantage over classical CG is its ability to use an indefinite preconditioner and still to maintain the minimization properties. This advantage does not come without a price: while CG needs to store 5 vectors, and do 5 vector operations per iteration, IP-CG needs to store 7 vectors, and do 13 vector operations per iteration.

IP-CG's advantage over GMRES is the fact that it uses a Lanczos iteration, so it does not need to store all the basis. Its advantege over QMR and BiCGStab is that it minimizes a real norm of the error. Another potential advantage of IP-CG over GMRES and QMR is the ability to base the stopping criteria on an estimate of the A-norm of the error. Indeed, the Hestenes-Stiefel estimate in classical CG can be easily incorporated in IP-CG. More advanced methods have been proposed (see, for example, [8, 1]), and maybe some of them are usable in IP-CG.

4. Indefinitely Preconditioned MINRES

The MINRES algorithm can be used to solve $Ax = b$ for any Hermitian matrix, and a preconditioner can be used as long as it is Hermitian positive definite. In this section we will show a variant of MINRES that requires the opposite: any Hermitian preconditioner can be used as long as the matrix is positive definite.

Suppose that A and it is Hermitian positive definite, and that the preconditioner M is Hermitian. Like the algorithm used in Section 3, we use the A-conjugate Lanczos iteration on $M^{-1}A$ and $M^{-1}b$. We have found a matrix T_n and a basis Q_n to $\mathcal{K}_n = \mathcal{K}_n(M^{-1}A, M^{-1}b)$ with $M^{-1}AQ_n = Q_{n+1}\tilde{T}_n$ and $Q_n^*AQ_n = I_{n\times n}$. A is a Hermitian positive definite matrix, so there exists a lower triangular matrix L such that $A = LL^*$. We do not need to compute L, we use it only for the derivation of the algorithm. We will now show how how Q_n and $\tilde{T}_n$ can be used to solve the equation $L^*M^{-1}Ax = L^*M^{-1}b$, which has exactly the same solution as $Ax = b$.

Denote $\hat{Q}_n = L^*Q_n$. The equation $Q_n^*AQ_n$reduces to $\hat{Q}_n^*\hat{Q}_n = I_{n\times n}$, so $\hat{Q}_n$ is a unitary matrix. Every $x \in \mathcal{K}_n$ can be written as $x = Q_n y$, so we have

$$\begin{aligned}
\min_{x\in\mathcal{K}_n} \|L^*M^{-1}Ax - L^*M^{-1}b\|_2 &= \min_y \|L^*M^{-1}AQ_n y - L^*M^{-1}b\|_2 \\
&= \min_y \|L^*Q_{n+1}\tilde{T}_n y - L^*M^{-1}b\|_2 \\
&= \min_y \|\hat{Q}_{n+1}\tilde{T}_n y - L^*M^{-1}b\|_2 \\
&= \min_y \|\tilde{T}_n y - \hat{Q}_{n+1}^*L^*M^{-1}b\|_2 \\
&= \min_y \|\tilde{T}_n y - Q_{n+1}^*LL^*M^{-1}b\|_2 \\
&= \min_y \|\tilde{T}_n y - Q_{n+1}^*AM^{-1}b\|_2 \\
&= \min_y \|\tilde{T}_n y - \|M^{-1}b\|_A e_1\|_2
\end{aligned}$$

We can iteratively find solutions y_n to $\min_y \|\tilde{T}_n y - \|M^{-1}b\|_A e_1\|_2$ and form $x_n = Q_n y_n$ in the same way as it is done in MINRES. As we can see we do not have to actually use L. We only rely on its existence. We include the pseudo-code for the new algorithm in the Appendix (Algorithm 3). We refer to this algorithm as IP-MINRES from here on.

A different, more technical way, to derive IP-MINRES is to start from regular write the equations for MINRES on $L^*M^{-1}Ly = L^*M^{-1}b$ and multiply all vectors generated by the iteration by L^{-*}. The matrix L will disappear from the equations and we will get Algorithm 3. In order to streamline this paper we will not give the details of this derivation.

5. Reformulation of older methods

The U-conjugate iterations can be used to give new formulation to older, well-known, algorithms. The purpose of this section is to show that these algorithms are similar to the ones presented in the previous sections, and that the framework presented in this paper are useful in building Krylov-subspace methods.

5.1. **Preconditioned MINRES using M^{-1}-Conjugate Lanczos.** If we A is Hermitian and M is positive definite we can use regular MINRES. The MINRES iteration basically does a U-conjugate Lanczos iteration on AM^{-1} with $U = M^{-1}$. The matrix $UA = M^{-1}AM^{-1}$ is Hermitian so this is indeed a Lanczos process. We can use the same techniques presented in Section 4 to solve the equation $Ax = b$. This is exactly what preconditioned MINRES does.

5.2. **Minimum Residual using A^*A-Conjugate Lanczos and Arnoldi.** If A is not Hermitian or Hermitian indefinite we cannot use $U = A$. Instead we can use $U = A^*A$. After iteration n we will have an A^*A-conjugate basis to the Krylov subspace $\mathcal{K}_n(A, b)$. We can use the Conjugate Directions method to produce an optimal A^*A-norm approximation, that is $\|e_n\|_{A^*A}$ is minimized. Note that

$$\begin{aligned} \|e_n\|_{A^*A} &= \sqrt{(x_n - x)^* A^* A (x_n - x)} \\ &= \|A(x_n - x)\|_2 \\ &= \|r_n\|_2 \end{aligned}$$

so the resulting algorithm finds the minimum residual solution in $\mathcal{K}(A, b)$. When viewed in that way we see that this is a different formulation of the GMRES algorithm. A preconditioner can be added quite easily by applying the iteration on $M^{-1}A$ and $M^{-1}b$ but keeping $U = A^*A$. We can carefully avoid the need to apply A^* and to do only one multiplication by A or solve for M. Unfortunately, we do need to keep two set of vectors, not just one as is required in the classical version of GMRES. Unlike GMRES this algorithm minimizes the residual of the original system, not the preconditioned system.

If A is Hermitian, then $UA = A^3$ which is a Hermitian matrix. In this case we can use the U-conjugate Lanczos iteration, and our algorithm reduces to a new version of MINRES. This version is possibly more stable than regular MINRES because we use the basis vectors in a CG-like process, instead of solving a least-squares problem. The main drawback of this method is that it cannot be preconditioned easily. Simply applying M^{-1} will not keep the symmetry of the iteration. To precondition with M we will need a factorization $M = LL^T$ and solve the equation $L^{-1}AL^{-T}(L^T y) = L^{-1}b$. There is no way to eliminate L from the iteration. Another disadvantaged is that we cannot use an indefinite preconditioner.

6. Numerical experiments and discussion

6.1. **Setup.** We have implemented the IP-CG and IP-MINRES compared them to older algorithms (GMRES, QMR, Bi-CGSTAB and CG). All the precondtioners (definite or indefinite) were built using WSMP [10]. We also used the implementation of GMRES, QMR, Bi-CGSTAB and CG in that library. We stop the iterative method and declare convergence after the relative residual has dropped below 10^{-11}(preconditioned GMRES might do a few more iterations then is actually needed because the algorithm keeps track of the preconditioned residual, and checks the actual residual only before declaring convergence). Running times were measured on a 2.13 GHz Intel Core 2 Duo computer with 4 GB of main memory, running Linux 2.6. This computer has 2 processors, but our solver only uses one. All experiments are done in 64-bit mode.

Table 1 lists the SPD matrices used to test the indefinitely preconditioned solvers, along with their kind and sizes in terms of both dimension and the number of nonzeros. The matrices were obtained from the University of Florida sparse matrix collection [4].

6.2. **Indefinite preconditioner.** In this section we list and analyze the results for instances where the preconditioner was indefinite. On all the instances listed in Table 2 MATLAB's [13] CHOLINC function produced a triangular factor with a negative value on the diagonal. This indicates that during the incomplete factorization the matrix becomes indefinite.

TABLE 1. Test matrices

Matrix	N	NNZ	Kind
ROTHBERG/CFD1	70,656	1,825,580	CFD problem
ROTHBERG/CFD2	123,440	3,085,406	CFD problem
GHS_PSDEF/VANBODY	47,072	2,329,056	Structural problem
BOEING/PWTK	217,918	11,524,432	Structural problem
INPRO/MSDOOR	415,863	19,173,163	Structural problem
ND/ND24K	72,000	28,715,634	2D/3D problem
DNVS/X104	108,384	8,713,602	Structural problem
SCHENK_AFE/AF_SHELL7	504,855	17,579,155	Structural problem
GHS_PSDEF/BMWCRA_1	148,770	10,641,602	Structural problem
GHS_PSDEF/LDOOR	952,203	42,493,817	Structural problem
GHS_PSDEF/OILPAN	73,752	2,148,558	Structural problem
WISSGOTT/PARABOLIC_FEM	525,825	3,674,625	CFD problem
DNSV/SHIPSEC5	179,860	4,598,604	Structural problem
DNVS/SHIP_003	121,728	3,777,036	Structural problem

We compare IP-CG and IP-MINRES to GMRES (without restarts and with restarts after 60 iterations), the symmetric variant of QMR and to Bi-CGSTAB. Theoretically, GMRES is the optimal algorithms since it finds the minimum residual solution, but it doesn't use a short recurrence. Symmetric QMR and Bi-CGSTAB are sub-optimal (for example, QMR minimizes a quasi-norm and not the real norm), but they use a short recurrence. IP-CG and IP-MINRES bridge the gap: they are theoretically optimal and they use a short recurrence.

The results appear in Table 2. The results show that our new algorithms converge when the preconditioner is indefinite, and that IP-CG is indeed a more robust version of CG. As long as there are no restarts in all but one instance GMRES does less iterations and converges faster. The comparison between IP-MINRES and GMRES is especially interesting: theoretically both algorithms are equivalent, but GMRES does less iterations. This suggests that there are stability issues when using a short recurrence. The new algorithms do less operations-per-iteration than GMRES, but that is negligible when a preconditioner is used because the running time is dominated by the cost of applying the preconditioner.

IP-CG and IP-MINRES are usually faster than QMR, but only marginally. IP-MINRES is theoretically superior to QMR since it minimizes the 2-norm of the residual, not a quasi-norm like QMR does. It should be noted that IP-CG and IP-MINRES are more robust than QMR since they cannot breakdown (divide by zero), like QMR can. A robust implementation of QMR needs to incorporate look aheads. The implementation of symmetric QMR that we use does not use look aheads. Both algorithms are faster then BiCGSTab on all instances.

The new algorithms also use less memory, so for memory-stressed scenarios (for example: solving a very large matrix, or solving several matrices concurrently) they allow a denser preconditioner. The "Precond Density" column was added in order to explore this issue. The value in the density column is the ratio between the number of non-zeros in the incomplete factor and the number of rows in the matrix, that is the number of non-zeros required to store the incomplete factor is density $\times$ #rows. For a restart value of k GMRES needs to store k complete vectors, so a total of $k \times$ #rows non-zeros. Therefore, if we wish to compare the amount of memory used to store the preconditioner to the number of non-zeros to store the vectors in GMRES (and is not needed in IP-CG) we need to compare k and density. If we examine the results for the two instances of MSDOOR, we see that IP-CG with the denser preconditioner (droptol $= 2 \times 10^{-4}$)

TABLE 2. Running time and number of iterations for instances in Table 1 where the preconditioner is indefinite. Density is the average amount of non-zeros per column in the incomplete factor. Preconditioner density is the ratio between the number of nonzeros in the incomplete factor and the number of rows in the matrix.

Matrix	Droptol	Precond Density	IP MINRES (Alg 3)	IP-CG (Alg 2)	GMRES(60)	GMRES	QMR	BiCGSTAB
CFD1	2×10^{-3}	197	127 its 23 sec	125 its 22 sec	117 its 23 sec	77 its 19 sec	139 its 24 sec	165 its 43 sec
CFD2	2×10^{-3}	258	161 its 51 sec	160 its 51 sec	87 its 37 sec	64 its 32 sec	174 its 53 sec	237 its 112 sec
VANBODY	2×10^{-3}	124	84 its 5.7 sec	85 its 5.8 sec	48 its 5.5 sec	48 its 5.5 sec	87 its 5.8 sec	117 its 11.4 sec
PWTK	2×10^{-3}	177	97 its 38 sec	98 its 38 sec	104 its 41 sec	71 its 34 sec	99 its 38 sec	94 its 57 sec
MSDOOR	8×10^{-4}	136	327 its 145 sec	336 its 146 sec	358 its 179 sec	108 its 78 sec	338 its 146 sec	610 its 444 sec
MSDOOR	2×10^{-4}	139	36 its 41 sec	36 its 41 sec	29 its 39 sec	29 its 39 sec	36 its 41 sec	40 its 55 sec
ND24K	4×10^{-4}	700	218 its 155 sec	217 its 154 sec	179 its 145 sec	83 its 118 sec	270 its 169 sec	592 its 416 sec
X104	2×10^{-2}	168	67 its 22 sec	66 its 22 sec	45 its 19 sec	45 its 19 sec	64 its 22 sec	90 its 38 sec
X104	2×10^{-3}	178	20 its 15 sec	20 its 15 sec	18 its 15 sec	18 its 15 sec	20 its 15 sec	15 its 17 sec
LDOOR	2×10^{-3}	122	59 its 69 sec	62 its 69 sec	59 its 75 sec	59 its 75 sec	66 its 71 sec	39 its 77 sec

uses less memory and is faster than GMRES with any reasonable restart value with a sparser preconditioner (droptol $= 8 \times 10^{-4}$). This is also true for the two instances of X104.

We explore this issue further in Table 3. In this set of experiments we have taken the largest matrix in our suite, ND24K, and solve it using different drop-tolerance values. The results show that GMRES is faster than IP-CG, but if we want to examine what can happen on a memory-tight situation we should compare the "density" column to the restart value. From Table 3 we see that the minimum amount storage for non-zeros needed by GMRES to solve the system in reasonable time is $749 \times$ #rows(drop-tolerance 5×10^{-4}). The minimum amount of memory needed by IP-CG is $631 \times$ #rows. The difference $118 \times$ #rows can be the difference between being able to solve the matrix on a given machine, or not.

6.3. **Positive definite preconditioner.** In this section we list and analyze the results for instances where the preconditioner was definite. On all the instances listed in Table 4 the preconditioner is definite. We compare IP-CG and IP-MINRES to CG and to GMRES (without restart). Usually, when both the matrix and the preconditioner are positive definite CG is used. Under exact arithmetic IP-CG is identical to CG. The goal of this set of experiments is to check whether under finite-accuracy arithmetic IP-CG's performance is similar to CG's. We also wish to check, using the comparison to GMRES, IP-MINRES's sensitivity to numerical instabilities.

The results appear in Table 4. The results show that indeed IP-CG acts very similar to CG and converges at about the same number of iterations (with cases of slight advantage to both algorithms). CG performs fewer operations per iteration, so it is a bit faster. Nevertheless,

TABLE 3. Detailed results for matrix ND24K.

Droptol	Precond Density	IP-CG (Alg 2)	GMRES	GMRES(120)	GMRES(200)
8×10^{-4}	574	FAIL	439 its 188 sec	FAIL	FAIL
7×10^{-4}	528	FAIL	338 its 146 sec	FAIL	2000 its 569 sec
6×10^{-4}	553	FAIL	396 its 181 sec	FAIL	FAIL
5×10^{-4}	631	582 its 233 sec	118 its 116 sec	118 its 116 sec	118 its 116 sec
4×10^{-4}	700	217 its 154 sec	83 its 118 sec	83 its 118 sec	83 its 118 sec
3×10^{-4}	719	192 its 156 sec	78 its 128 sec	78 its 128 sec	78 its 128 sec
2×10^{-4}	792	141 its 167 sec	60 its 143 sec	60 its 143 sec	60 its 143 sec
1×10^{-4}	854	46 its 209 sec	35 its 207 sec	35 its 207 sec	35 its 207 sec

TABLE 4. Running time and number of iterations for instances in Table 1 where the preconditioner is definite.

Matrix	Droptol	Precond Density	IP MINRES (Alg 3)	IP-CG (Alg 2)	CG	GMRES
AF_SHELL7	2×10^{-3}	97	128 its 59 sec	137 its 60 sec	137 its 57 sec	131 its 81 sec
BMWCRA_1	2×10^{-3}	215	128 its 59 sec	137 its 60 sec	137 its 57 sec	147 its 61 sec
LDOOR	2×10^{-4}	122	17 its 57 sec	16 its 56 sec	17 its 56 sec	18 its 58 sec
OILPAN	8×10^{-4}	89	39 its 3.8 sec	39 its 3.7 sec	39 its 3.5 sec	39 its 3.6 sec
PARABOLIC_FEM	2×10^{-3}	19	68 its 13.7 sec	73 its 13.6 sec	73 its 11.6 sec	70 its 19.3 sec
SHIPSEC5	2×10^{-3}	95	45 its 11.4 sec	46 its 11.3 sec	45 its 10.7 sec	47 its 12.3 sec
SHIP_003	2×10^{-3}	108	84 its 13.1 sec	85 its 13.0 sec	89 its 12.7 sec	87 its 15.5 sec

IP-CG is more robust, being able to handle an indefinite preconditioner, so the user can trade a few percents of performance for increased robustness.

The comparison of IP-MINRES and IP-CG to GMRES show that the numerical instabilities encountered when using an indefinite preconditioner do no longer appear when the preconditioner is definite. In most cases IP-MINRES does less iterations then GMRES and it is faster. We discuss this issue further in section 6.5.

TABLE 5. Comparing strategies: using an indefinite preconditioner or forcing definiteness.

Matrix	Droptol	IP-CG (Alg 2)	CG, run 1	CG, run 2
CFD1	4×10^{-4}	126 its	$\alpha = 0.5$ 357 its	$\alpha = 0.01$ 64 its
OILPAN	2×10^{-5}	202 its	$\alpha = 0.5$ 889 its	$\alpha = 0.01$ 133 its
SHIP_003	2×10^{-4}	251 its	$\alpha = 0.5$ 1136 its	$\alpha = 0.01$ 247 its

6.4. **Using an indefinite preconditioner vs. forcing definiteness.** An alternative to using an indefinite preconditioner is to somehow to force the incomplete factorization to produce a definite preconditioner. A detailed experimental study of which strategy is better is out of the scope of this paper. The goal of this set of experiment is to show that in some cases it may be preferable to use an indefinite preconditioner.

There are many methods by which definiteness can be forces, see [3] for a survey. We have chosen to test one of these methods. More specifically, we chose to try the method suggested by Manteuffel in [12]. This method tries to find a value α such that the incomplete factorization of $\hat{A} = A + \alpha \text{diag}(A)$ is positive definite, and uses that factor as a preconditioner. The value of α is found using a trial-and-error method that can be quite expensive. Obviously, the quality of the preconditioner depends on the value of α that was used.

For our comparison we decided not to use trial-and-error method due to its cost. Instead, we chose to try two values for α, a small value and a large value, for every matrix in this set of experiments. The experiments in this section we done using MATLAB, and the preconditioner was built using MATLAB's CHOLINC function, so we report only the number of iterations. We stop the iterative method and declare convergence after the relative residual has dropped below 10^{-6}.

The results appear in Table 5. As can be seen from this table there are values of α which produce a better preconditioner, and there are values of α which produce a lower quality preconditioner. This demonstrates the effectiveness of our new methods, in that they provided reasonable results without a tuning parameter.

6.5. **Numerical stability: full conjugation vs. local conjugation.** The results in Table 2 indicate that the new solvers do not fulfill their potential when the preconditioner is indefinite: they should do about the same number of iterations as GMRES. It seems that this is not true for a definite preconditioner (Table 4). A natural suspect for the gap between the theoretical behavior and the actual behavior is the Lanczos process, which it's known instablilty causes the vectors to lose orthogonality. See §4in [9] for a discussion on the loss of orthogonlity in the Lanczos process and its effect on CG and MINRES.

To check this issue we compared the number of iterations when using a full conjugation to the number of iterations when using a local conjugation, that is using U-conjugate Arnoldi instead of U-conjugate Lanczos. Mathematically, U-conjugate Lanczos is sufficient, but the vectors may lose their U-orthogonality. In Table 6 we compare IP-CG its Arnoldi equivalent (FULL IP-CG) and IP-MINRES to GMRES (which is theoretically equivalent).

From the results we see that many times a long recurrence needs considerably less iterations. Other times, the short recurrence works equally as well as the long recurrence. This indicates that the usage of a short recurrence can cause numerical problems. The experiments also show that the problem is not directly connected to the use of an indefinite preconditioner: we have cases where the problem manifests for a definite preconditioner (CFD1-4.5×10^{-4}, OILPAN-NO PRECOND)

TABLE 6. Numerical stability: comparing full conjugation to local conjugation. In the OILPAN (NO PRECOND) instance convergence threshold was set to 10^{-5}.

Matrix	Droptol	Precond Definite?	IP-CG	FULL IP-CG	IP-MINRES	GMRES	CG
CFD1	2×10^{-3}	NO	125 its	77 its	127 its	77 its	N/A
CFD1	4.5×10^{-4}	YES	85 its	69 its	84 its	69 its	85 its
CFD1	2×10^{-3}	YES	48 its	47 its	48 its	46 its	48 its
OILPAN	NO PRECOND	N/A	783 its	747 its	297 its	242 its	783 its
OILPAN	8×10^{-3}	NO	441 its	142 its	437 its	130 its	N/A
OILPAN	1.5×10^{-3}	NO	63 its	58 its	64 its	51 its	N/A
OILPAN	8×10^{-4}	YES	39 its	42 its	39 its	36 its	39 its
PWTK	4×10^{-3}	NO	149 its	103 its	149 its	103 its	N/A
PWTK	1×10^{-3}	NO	77 its	55 its	77 its	55 its	N/A
PWTK	8×10^{-4}	YES	61 its	55 its	61 its	54 its	61 its

and cases where manifests very weakly for an indefinite preconditioner (OILPAN-1.5×10^{-3}). There are cases where CG converges slower than it should even though the preconditioner is definite, so apparently both IP-CG and CG suffer from the same numerical instability. There seems to be a connection between the quality of the preconditioner and numerical instability encountered. Indefinite incomplete factorization tend to be lower quality preconditioners because the indefiniteness in the incomplete factors indicate a the incomplete factorization drops nonzeros too aggressively.

7. Conclusion and Open Questions

We have presented new versions of CG and MINRES that accept a Hermitian indefinite preconditioner as long as the preconditioned matrix is Hermitian positive definite. The motivation for the new algorithm is the possible failure of incomplete factorization to produce a positive definite preconditioners. We have conducted extensive numerical experiments and have demonstrated the robustness and the utility of our new algorithms in some cases.

The experiments also demonstrate that the new algorithms do not fulfill their full theoretical potential: GMRES converges in less iterations. The experiments hint that the problem is caused by numerical instabilities in the Lanczos process, and that CG suffers from the same problem. A possible strategy to improve the stability is to reorthogonlize the vectors periodically, when they lose the A-orthogonality too much. To do so, the intermediate vectors must be kept (like GMRES does), but if the reorthogonlization process is infrequent enough they can be kept in secondary storage. Find a method that balances between keeping stability and avoiding reorthogonlization can be challenging. Such techniques have been employed in eigensolvers (see §5.3 in [19]). Another technique that might help is using a coupled two-term recurrence instead of the three-term recurrence we currently use. This technique was used to improve the numerical behavior of QMR in [7].

Another interesting question that arises from this paper is whether it is better to use the incomplete factorization process as-is, even if the preconditioner turns out to be indefinite, or to use incomplete factorization methods that guarantee a positive definite preconditioner ? To answer this question a comprehensive experimental study would be necessary, since there many methods to guarantee positive definiteness (see [3]). Obviously, improving the stability of the

short recurrence algorithms can affect the answer to this question. Such an improvement can affect the convergence of both the definite and indefinite preconditioner case, but the improvement may not be the same for both approaches.

References

[1] Mario Arioli. A stopping criterion for the conjugate gradient algorithm in a element method framework. Technical report, Numerische Mathematik, 2000.

[2] Haim Avron, Esmond Ng, and Sivan Toledo. Using perturbed QR factorizations to solve linear least-squares problems. Accepted to *SIAM Journal on Matrix Analysis and Applications*, 22 pages, 2008.

[3] Michele Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[4] T. Davis. The University of Florida sparse matrix collection. *http://www.cise.ufl.edu/research/sparse/matrices*.

[5] V. Faber and T.A. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21(2):352–362, 1984.

[6] Roland W. Freund and Noël M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60(1):315–339, Dec 1991.

[7] Roland W. Freund and Noël M. Nachtigal. An implementation of the QMR method based on coupled two-term recurrences. *SIAM J. Sci. Comput.*, 15(2):313–337, 1994.

[8] Gene H. Golub. Matrices, moments and quadrature II; how to compute the norm of the error in iterative methods. *BIT*, 37:687–705, 1997.

[9] Anne Greenbaum. *Iterative methods for solving linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[10] A. Gupta. WSMP: Watson sparse matrix package (Part-III: iterative solution of sparse systems). Technical Report RC-24398, IBM T.J. Watson Research Center, Yorktown Heights, NY, 2007.

[11] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, Dec 1952.

[12] T.A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comput.*, 34(473), 1980.

[13] The MathWorks. Matlab version 7.2. software package, January 2006.

[14] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12:617–629, 1975.

[15] Christopher C. Paige and Michael A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, 1982.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[17] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[18] J. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, 1994.

[19] G. W. Stewart. *Matrix algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[20] H. A. van der Vorst. BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.

Appendix: Pseudo-code for the new algorithms

Algorithm 2 Indefinitely Preconditioned CG (IP-CG)

Input: Hermitian positive definite A, a right hand side b and an Hermitian preconditioner M
$q_1 = M^{-1}b$
$l_1 = Aq_1$
$w = \sqrt{q_1^* l_1}$
$l_1 = l_1/w$
$q_1 = q_1/w$
$r^{(0)} = b - Ax$
$x^{(0)} = 0$
for $t = 1, 2, \ldots$
  $\gamma_t = q_t^* r^{(t-1)}$
  $x^{(t)} = x^{(t-1)} + \gamma_t q_t$
  $r^{(t)} = r^{(t-1)} - \gamma_t l_t$
  check for convergence
  $v_{t+1} = M^{-1} l_t$
  $H_{t,t} = l_t^* v_{t+1}$
  $H_{t-1,t} = H_{t,t-1} (= l_{t-1}^* v_{t+1})$
  $q_{t+1} = v_{t+1} - H_{t,t} q_t - H_{t-1,t} q_{t-1}$
  $l_{t+1} = Aq_{t+1}$
  $H_{t+1,t} = \sqrt{q_{t+1}^* l_{t+1}}$
  $l_{t+1} = l_{t+1}/H_{t+1,t}$
  $q_{t+1} = q_{t+1}/H_{t+1,t}$
end for

Algorithm 3 Indefinitely Preconditioned MINRES (IP-MINRES)

Input: Hermitian positive definite A, a right hand side b and an Hermitian preconditioner M
$q_1 = M^{-1}b$
$l_1 = Aq_1$
$w_1 = \sqrt{q_1^* l_1}$
$l_1 = l_1/w_1$
$q_1 = q_1/w_1$
$r^{(0)} = b - Ax$
$x^{(0)} = 0$
$s_{-2} = 0, s_{-1} = 0$
for $t = 1, 2, \ldots$ until convergence
  $v_{t+1} = M^{-1}l_t$
  $H_{t,t} = l_t^* v_{t+1}$
  $H_{t-1,t} = H_{t,t-1} (= l_{t-1}^* v_{t+1})$
  $q_{t+1} = v_{t+1} - H_{t,t}q_t - H_{t-1,t}q_{t-1}$
  $l_{t+1} = Aq_{t+1}$
  $H_{t+1,t} = \sqrt{q_{t+1}^* l_{t+1}}$
  $l_{t+1} = l_{t+1}/H_{t+1,t}$
  $q_{t+1} = q_{t+1}/H_{t+1,t}$
  $U_{t-2,t} = s_{t-2}H_{t-1,t}$
  if $(t > 2)$ $U_{t-1,t} = c_{t-2}H_{t-1,t}$ else $U_{t-1,t} = H_{t-1,t}$
  if $(t > 1)$ $U_{t,t} = -s_{t-1}U_{t-1,t} + c_{t-1}H_{t,t}$ else $U_{t,t} = H_{t,t}$
  $U_{t-1,t} = c_{t-1}U_{t-1} + s_{t-1}H_{t,t}$
  compute Givens rotation factors c_t and s_t on $\begin{bmatrix} U_{t,t} & H_{t+1,t} \end{bmatrix}^T$
  $U_{t,t} = c_t U_{t,t} + s_t H_{t+1,t}$
  $w_{t+1} = -s_t w_t$
  $w_t = c_t w_t$
  $m_t = (U_{t,t})^{-1}(q_t - U_{t-1,t}m_{t-1} - U_{t-2,t}m_{t-2})$
  $x^{(t)} = x^{(t-1)} + w_t m_t$
end for

Current address: Haim Avron: Tel-Aviv University and IBM T.J. Watson Research Center (summer intern), Sivan Toledo: Tel-Aviv University and MIT, Anshul Gupta: IBM T. J. Watson Research Center