Improving Triangular Preconditioner Updates for Nonsymmetric Linear Systems

Jurjen Duintjer Tebbens and Miroslav Tůma

Institute of Computer Science, Czech Academy of Sciences, Pod Vodárenskou věží 2, 18207 Praha 8, Czech Republic
{tebbens,tuma}@cs.cas.cz

Abstract. We present an extension of an update technique for preconditioners for sequences of non-symmetric linear systems that was proposed in [5]. In addition, we describe an idea to improve the implementation of the update technique. We demonstrate the superiority of the new approaches in numerical experiments with a model problem.

1 Introduction

Sequences of linear systems with large and sparse matrices arise in many applications like computational fluid dynamics, structural mechanics, numerical optimization as well as in solving non-PDE problems. In many cases, one or more systems of *nonlinear* equations are solved by a Newton or Broyden-type method [6], and each nonlinear equation leads to a sequence of linear systems. The solution of sequences of linear systems is the main bottleneck in many of the above mentioned applications. For example, some solvers need strong preconditioners to be efficient and computing preconditioners for individual systems separately may be very expensive.

In recent years, a few attempts to update preconditioners for sequences of large sparse systems have been made. If a sequence of linear systems arises from a quasi-Newton method, straightforward approximate small rank updates can be useful (this has been done in the SPD case in [9,3]). For shifted SPD linear systems, an update technique was proposed in [8] and a different one can be found in [2]. The latter technique, based on approximate diagonal updates, has been extended to sequences of parametric complex symmetric linear systems (see [4]). This technique, in turn, was generalized to approximate (possibly permuted) triangular updates for nonsymmetric sequences [5]. In addition, recycling of Krylov subspaces by using adaptive information generated during previous runs has been used to update both preconditioners and Krylov subspace iterations (see [7,10,1]).

In this paper we address two ways to improve the triangular updates of preconditioners for nonsymmetric sequences of linear systems from [5]. It was discussed in [5] that triangular updates may be particularly beneficial under three types of circumstances: first, if preconditioner recomputation is for some reason expensive (e.g. in parallel computations, matrix-free environment); second,

I. Lirkov, S. Margenov, and J. Waśniewski (Eds.): LSSC 2007, LNCS 4818, pp. 737–744, 2008.

if recomputed preconditioners suffer from instability and the updates relate to a more stable reference factorization; third, if the update is dominant, at least structurally, that is, if it covers a significant part of the difference between the current and reference matrix. Our first contribution is motivated by the third case. The updates from [5] exhibit this property whenever a permutation can be found such that one triangular part of the permuted difference matrix clearly dominates the other part. Experiments given there show that this is the case in many types of applications where the permutation may not even be needed. Nevertheless, these techniques neglect one of the two triangular parts of the (permuted) difference matrix and possibly useful information contained in this part is lost. We will describe here how both triangular parts can be taken into account by considering a simple but effective extension of the original technique. We compare the new idea with the original strategy and experiments demonstrate its improved power on a model problem. Our second contribution is of a more technical nature. We present a different implementation for the triangular updates. The important time savings with respect to the strategy used in [5] confirm what we only assumed there, and reveal more about the potential of the preconditioner updates.

In the next section we address the first improvement and Section 3 describes the second one. Numerical experiments are presented in Section 4. We denote by $\|\cdot\|$ an arbitrary, unspecified matrix norm.

2 Gauss-Seidel Type Updates

We consider a system $Ax = b$ with a factorized preconditioner $M = LDU$ and let $A^+x^+ = b^+$ be a system of the same dimension, and denote the difference matrix $A - A^+$ by B . We search for an updated preconditioner M^+ for $A^+x^+ = b^+$. We have $\|A - M\| = \|A^+ - (M - B)\|$, hence the norm of the difference $A^+ - M^+$ with $M^+ \equiv M - B$, called the *accuracy* of M^+ (with respect to A^+), is the same as that of M with respect to A. If $M^+ = M - B$ is the preconditioner, we need to solve systems with $M - B$ as system matrix in every iteration of the iterative solver. Clearly, for general B the preconditioner $M^+ = M - B$ cannot be used in practice since the systems are too expensive to solve. Instead, we will consider cheap approximations of $M - B$. If $M - B$ is nonsingular, we approximate it by a product of factors which are easier to invert. The approximation consists of two steps. First, we approximate $M - B$ as

$$M - B = L(DU - L^{-1}B) \approx L(DU - B), \tag{1}$$

or by

$$M - B = (LD - BU^{-1})U \approx (LD - B)U. \tag{2}$$

The choice between (1) and (2) is based on the distance of L and U to identity. If $\|I - L\| < \|I - U\|$ then we will base our updates on (1) and in the following C will denote the matrix $DU - B$. If on the other hand $\|I - L\| > \|I - U\|$, then we will use (2) and we define C by $C \equiv LD - B$. Our implementation chooses the appropriate strategy adaptively.

Our next goal is to find an approximation of C that can be used as a preconditioner. We split C as $C = L_C + D_C + U_C$, where L_C, D_C, U_C denote the strict lower triangular, the main diagonal and the strict upper triangular part of C, respectively. In [5], C was approximated by a single triangular factor $L_C + D_C$ or $U_C + D_C$. In this paper we propose a Gauss-Seidel type of approach that takes into account both triangular parts of C. We will use the classical symmetric Gauss-Seidel approximation $C \approx (L_C + D_C)D_C^{-1}(U_C + D_C)$. Putting the two approximation steps together, we obtain an updated preconditioner of the form

$$M^+ = L(L_C + D_C)D_C^{-1}(U_C + D_C), \quad C = DU - B, \tag{3}$$

when $\|I - L\| < \|I - U\|$, and otherwise we use

$$M^+ = (L_C + D_C)D_C^{-1}(U_C + D_C)U, \quad C = LD - B. \tag{4}$$

These updates can be cheaply obtained. C is a difference of two sparse matrices, the splitting of C is trivial. The updated preconditioner has one additional factor compared with the original factorization LDU, hence its application is a little more expensive. In cases where the sparsity patterns of B and L or U differ significantly, the solves with the updated factors are also more expensive than with the original factors. The choice between (3) and (4) can be based on comparing the Frobenius norms of $I - L$ and $I - U$, which is very cheap with sparse factors. As for storage costs, the original factorization and the reference matrix A must be available when applying updates of this form.

We showed in [5] that the accuracy of the updates introduced there increases with a factor L (or U) closer to identity and with a smaller error $\|C - D_C - U_C\|$ (or $\|C - D_C - L_C\|$). Also stability increases with these properties. Our theoretical results explained why the updates are often more powerful than old factorizations and may even be, in favorable cases, more powerful than newly computed factorizations. For the updates introduced in this paper, similar results hold. We will now concentrate on comparison of the new Gauss-Seidel updates with the original technique. The updates from [5] can be written as

$$M^+ = L\,(D_C + U_C)\,, \quad C = DU - B, \tag{5}$$

and

$$M^+ = (L_C + D_C)\,U, \quad C = LD - B. \tag{6}$$

Intuitively it is clear that the Gauss-Seidel type updates may be expected to be more powerful than (5) and (6) if their approximation of C is stronger than with one triangular part only. Let $E = A - LDU$ denote the accuracy error of the preconditioner for A and let $G = C - (L_C + D_C)D_C^{-1}(U_C + D_C) = L_C D_C^{-1} U_C$ be the approximation error of C for the Gauss-Seidel update. We split B as $B = L_B + D_B + U_B$, where L_B, D_B, U_B denote the strict lower triangular, the main diagonal and the strict upper triangular part of B, respectively. The accuracy of (6) can then be written as

$$\|A^+ - (L_C + D_C)U\| = \|A - LDU - B + (L_B + D_B)U\| = \|E - B(I - U) - U_B U\|,$$

where we used that $L_C + D_C = LD - L_B - D_B$. Similarly, the accuracy of (4) is

$$\begin{aligned}\|A^+ - (L_C + D_C)(D_C^{-1}U_C + I)U\| = \\ \|E - U_B - (L_B + D_B)(I - U) - (L_C D_C^{-1} + I)U_C U\| = \\ \|E - U_B - (L_B + D_B)(I - U) + U_B\, U - G \cdot U\| = \\ \|E - B(I - U) - G \cdot U\|.\end{aligned}$$

Hence the accuracies of (4) and (6) are of the form $\|X - GU\|$ and $\|X - U_B U\|$, where $X = E - B(I - U)$. As U_B is nothing but the error to approximate C according to (6), here we see the effect of the approximation errors G and U_B with the two update techniques. The updates (5) and (3) share a similar relation.

Now let us denote by *striu* the strict upper triangular part and by *tril* the lower triangular part of a matrix (including the main diagonal). In the Frobenius norm, denoted by $\|\cdot\|_F$, positive influence of the Gauss-Seidel technique can be expressed as follows. Assume that

$$\|tril(E - B(I - U) - G \cdot U)\|_F^2 + \tag{7}$$
$$\|striu(E - B + (B - G) \cdot U)\|_F^2 \leq \tag{8}$$
$$\|tril\,(E - B(I - U))\,\|_F^2 + \tag{9}$$
$$\|striu\,(E - B + (L_B + D_B)U)\,\|_F^2 \quad , \tag{10}$$

then

$$\begin{aligned}\|A^+ - (L_C + D_C)(D_C^{-1}U_C + I)U\|_F^2 = \\ \|tril(E - B(I - U) - G \cdot U)\|_F^2 + \|striu(E - B + (B - G) \cdot U)\|_F^2 \leq \\ \|tril\,(E - B(I - U))\,\|_F^2 + \|striu\,(E - B + (L_B + D_B)U)\,\|_F^2 = \\ \|A^+ - (D_C + L_C)\,U\|_F^2.\end{aligned}$$

The last equality follows from $striu(BU) = striu((U_B + L_B + D_B)U) = striu(U_B U) + striu((L_B + D_B)U) = U_B U + striu((L_B + D_B)U)$. Thus superiority of the Gauss-Seidel type update (6) may be expected if the contribution of $-GU$ to the lower triangular part of X reduces the entries of this part and if the contribution of $(B - G)U$ to the strict upper triangular part of X reduces the entries more than the contribution of $(L_B + D_B)U$ to this part. We will confirm exactly this behavior in the experiments in Section 4.

3 Alternative Implementation

The update (5) (or (6)) in [5] was implemented with two backward (or forward) solves. In the upper triangular case (5), the solve step with $D_C + U_C$, which is equal to $DU - (D_B + U_B)$, used separate loops with DU and with $D_B + U_B$ in [5]. These loops were tied together by scaling with the sum of the diagonal entries of DU and B. In detail: Let the entries of DU and B be denoted by $(du)_{ij}$ and b_{ij}, respectively, and consider a linear system $(DU - D_B - U_B)\,z = y$. Then for $i = n, n-1, \ldots, 1$ the subsequent cycles

$$z_i = y_i - \sum_{j>i} (du)_{ij} z_j, \qquad z_i = z_i - \sum_{j>i} b_{ij} z_j, \tag{11}$$

were used, followed by putting

$$z_i = \frac{z_i}{(du)_{ii} + b_{ii}}. \tag{12}$$

A first advantage of this implementation is that the solution process is straightforward. The sparsity patterns of DU and $D_B + U_B$, which are immediately available, do not need to be further processed. Another advantage of this implementation is that the difference matrix B may be sparsified in a different way for different matrices of a sequence. It was mentioned repeatedly in [5], however, that merging the two matrices DU and $D_B + U_B$ may yield better timings. Here we present results of experiments with merged factors which formed the sum $DU - D_B - U_B$, or its lower triangular counterpart, explicitly. This sum needs to be formed only once at the beginning of the solve process of the linear system, that is in our case, before the preconditioned iterations start. Every time the preconditioner is applied, the backward solve step with the merged factors may be significantly cheaper than with (11)–(12) if the sparsity patterns of DU and $D_B + U_B$ are close enough. In our experiments we confirm this.

4 Numerical Experiments

Our model problem is a two-dimensional nonlinear convection-diffusion model problem. It has the form (see, e.g. [6])

$$-\Delta u + Ru\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) = 2000x(1-x)y(1-y), \tag{13}$$

on the unit square, discretized by 5-point finite differences on a uniform grid. The initial approximation is the discretization of $u_0(x, y) = 0$. In contrast with [5] we use here $R = 100$ and different grid sizes. We solve the resulting linear systems with the BiCGSTAB [11] iterative method with right preconditioning. Iterations were stopped when the Euclidean norm of the residual was decreased by seven orders of magnitude. Other stopping criteria yield qualitatively the same results.

In Table 1 we consider a 70×70 grid, yielding a sequence of 13 matrices of dimension 4900 with 24220 nonzeros each. We precondition with ILU(0), which has the same sparsity pattern as the matrix it preconditions. This experiment was performed in Matlab 7.1. We display the number of BiCGSTAB iterations for the individual systems and the overall time to solve the whole sequence. The first column determines the matrix of the sequence which is preconditioned. The second column gives the results when ILU(0) is recomputed for every system of the sequence. In the third column ILU(0) is computed only for the first system and reused (frozen) for the whole sequence. In the remaining columns this first factorization is updated. 'Triang' stays for the triangular updates from [5], that

Table 1. Nonlinear convection-diffusion model problem with n=4900, ILU(0)

ILU(0), psize ≈ 24000				
Matrix	Recomp	Freeze	Triang	GS
$A^{(0)}$	40	40	40	40
$A^{(1)}$	25	37	37	27
$A^{(2)}$	24	41	27	27
$A^{(3)}$	20	48	26	19
$A^{(4)}$	17	56	30	21
$A^{(5)}$	16	85	32	25
$A^{(6)}$	15	97	35	29
$A^{(7)}$	14	106	43	31
$A^{(8)}$	13	97	44	40
$A^{(9)}$	13	108	45	38
$A^{(10)}$	13	94	50	44
$A^{(11)}$	15	104	45	35
$A^{(12)}$	13	156	49	42
overall time	13 s	13 s	7.5 s	6.5 s

Table 2. Nonlinear convection-diffusion problem with n=4900: Accuracies and values (7)–(10)

i	$\|A^{(i)} - M_{GS}^{(i)}\|_F^2$	$\|A^{(i)} - M_{TR}^{(i)}\|_F^2$	value of (7)	value of (9)	value of (8)	value of (10)
1	852	857	*	*	*	*
2	938	1785	377	679	560	1105
3	1102	2506	373	843	729	1663
4	1252	3033	383	957	869	2076
5	1581	3975	432	1155	1149	2820
6	1844	4699	496	1303	1388	3395
7	2316	5590	610	1484	1706	4106
8	2731	6326	738	1631	1993	4695
9	2736	6372	735	1642	2002	4731
10	2760	6413	742	1650	2018	4763
11	2760	6415	742	1650	2018	4765
12	2760	6415	742	1650	2018	4765

is for adaptive choice between (5) and (6). The last column presents results for the Gauss-Seidel (GS) updates (3) and (4). The abbreviation 'psize' gives the average number of nonzeros of the preconditioners.

As expected from [5], freezing yields much higher iteration counts than any updated preconditioning. On the other hand, recomputation gives low iteration counts but it is time inefficient. The new GS strategy from Section 2 improves the power of the original triangular update. Table 2 displays the accuracies of (4) (here denoted by M_{GS}) and (6) (denoted by M_{TR}) in the Frobenius norm and the values of (7-10). These values reflect the efficiencies of the two updates

Table 3. Nonlinear convection-diffusion model problem with n=49729, ILUT(0.2/5)

ILUT(0.2/5), psize $\approx$ 475000, ptime $\approx$ 0.05				
Matrix	Recomp	Freeze	Triang	GS
$A^{(0)}$	113/2.02	113/2.02	113/2.02/2.02	113/2.02
$A^{(1)}$	119/2.06	112/1.94	104/1.95/1.81	122/2.26
$A^{(2)}$	111/1.94	111/1.95	104/1.91/1.78	100/1.84
$A^{(3)}$	94/1.66	115/2.00	92/1.64/1.45	96/1.77
$A^{(4)}$	85/1.44	116/2.00	92/1.77/1.55	90/1.67
$A^{(5)}$	81/1.45	138/2.44	93/1.73/1.47	83/1.55
$A^{(6)}$	72/1.28	158/2.75	101/1.89/1.63	85/1.59
$A^{(7)}$	72/1.28	163/2.86	101/1.91/1.59	92/1.69
$A^{(8)}$	78/1.36	161/2.84	94/1.77/1.53	82/1.48
$A^{(9)}$	72/1.23	159/2.72	92/1.72/1.73	80/1.55
$A^{(10)}$	73/1.27	153/2.66	97/1.91/1.61	82/1.48

and confirm the remarks made after (7-10). Note that the first update in this sequence is based on (3), resp. (5) and thus the values (7-10) do not apply here.

In Table 3 we use the grid size 223 and obtain a sequence of 11 linear systems with matrices of dimension 49729 and with 247753 nonzeros. The preconditioner is ILUT(0.2,5), that is incomplete LU decomposition with drop tolerance 0.2 and number of additional nonzeros per row 5. This experiment was implemented in Fortran 90 in order to show improvements in timings for the alternative implementation strategy discussed in Section 3. The columns contain the BiCGSTAB iteration counts, followed by the time to solve the linear system, including the time to compute the (updated or new) factorization. In the column 'Triang' the last number corresponds to the implementation with merged factors as explained above and 'ptime' denotes the average time to recompute preconditioners.

The benefit of merging is considerable. Still, even with this improved implementation, the Gauss-Seidel type of updates happens to be faster than the standard triangular updates for several systems of the sequence. As for the BiCGSTAB iteration counts, for the majority of the linear systems Gauss-Seidel updates are more efficient. We have included in this table the results based on recomputation as well. In contrast to the results of the previous example, the decomposition routines are very efficient and exhibit typically in-cache behaviour. Then they often provide the best overall timings. This does not need to be the case in other environments like matrix-free or parallel implementations, or in cases where preconditioners are computed directly on grids.

5 Conclusion

In this paper we considered new ways for improving triangular updates of factorized preconditioners introduced in [5]. We proposed a Gauss-Seidel type of approach to replace the triangular strategy, and we introduced a more efficient

implementation of adaptive triangular updates. We showed on a model nonlinear problem that both techniques may be beneficial. As a logical consequence, it seems worth to combine the two improvements by adapting the new implementation strategy for Gauss-Seidel updates. We expect this to yield even more efficient updates. For conciseness, we did not present some promising results with the Gauss-Seidel approach generalized by adding a relaxation parameter.

Acknowledgement.

This work was supported by the project 1ET400300415 within the National Program of Research "Information Society". The work of the first author is also supported by project number KJB100300703 of the Grant Agency of the Academy of Sciences of the Czech Republic.

References

1. Baglama, J., et al.: Adaptively preconditioned GMRES algorithms. SIAM J. Sci. Comput. 20, 243–269 (1998)
2. Benzi, M., Bertaccini, D.: Approximate inverse preconditioning for shifted linear systems. BIT 43, 231–244 (2003)
3. Bergamaschi, L., et al.: Quasi-Newton Preconditioners for the Inexact Newton Method. ETNA 23, 76–87 (2006)
4. Bertaccini, D.: Efficient preconditioning for sequences of parametric complex symmetric linear systems. ETNA 18, 49–64 (2004)
5. Tebbens, J.D., Tůma, M.: Efficient preconditioning of sequences of nonsymmetric linear systems. SIAM J. Sci. Comput. (to appear)
6. Kelley, C.T.: Iterative methods for linear and nonlinear equations. SIAM, Philadelphia (1995)
7. Loghin, D., Ruiz, D., Touhami, A.: Adaptive preconditioners for nonlinear systems of equations. J. Comput. Appl. Math. 189, 326–374 (2006)
8. Meurant, G.: On the incomplete Cholesky decomposition of a class of perturbed matrices. SIAM J. Sci. Comput. 23, 419–429 (2001)
9. Morales, J.L., Nocedal, J.: Automatic preconditioning by limited-memory quasi-Newton updates. SIAM J. Opt. 10, 1079–1096 (2000)
10. Parks, M.L., et al.: Recycling Krylov subspaces for sequences of linear systems. SIAM J. Sci. Comput. 28, 1651–1674 (2006)
11. van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. SIAM J. Sci. Stat. Comput. 12, 631–644 (1992)