On residual smoothing in ILUM-type preconditioning

Lutz Grosz[*]

(Received 7 August 2000)

Abstract

Incomplete block factorisations are used to construct flexible preconditioners for iterative linear solvers. In practice these approaches have shown to be very effective and robust. Especially they are more suitable for parallel computer architectures when comparing to classic ILU preconditioning. In this paper we introduce residual smoothing

[*]School of Mathematical Sciences, Australian National University, Canberra, ACT 0200, Australia. Current address: Institute for Information and Mathematical Sciences, Massey University at Albany, New Zealand. mailto:l.grosz@massey.ac.nz

[0]See http://anziamj.austms.org.au/V42/CTAC99/Gros for this article and ancillary services, © Austral. Mathematical Soc. 2000. Published 27 Nov 2000.

into the forward/backward substitution in order to compensate the element dropping in the Schur complement.

Contents

1 Introduction **C655**

2 Block Factorisation **C657**

3 Smoothing **C659**

4 Forward/backward Substitution **C662**

5 Examples **C662**

5.1 Convection Driven Diffusion C663

5.2 L-Shaped Domain C667

6 Conclusion **C668**

References **C669**

1 Introduction

We consider Krylov subspace methods [13] in order to solve the linear system

$$Ax = b\,, \tag{1}$$

where $A = (a_{i,j})_{i,j=1,n}$ is a real, non-singular, sparse coefficient matrix. One tries to improve the convergence rate and the robustness by introducing a preconditioner matrix M^{-1} in such a way that the new iteration matrix $M^{-1}A$ is close to the identity matrix I. The construction of M^{-1} as well as the evaluation of matrix-vector products with M^{-1} must not be too costly. The computer architecture used is an important factor in the efficiency of a preconditioner. For instance the classical ILU preconditioner is not suitable for parallel architectures. Block versions of ILU, like ILUM [9], BILUM [12] and AMLI [1], have a much better data flow for parallel architectures.

As an alternative to ILU, some authors propose the idea of constructing the matrix M^{-1} explicitly by minimising a norm of the defect $I - M^{-1}A$ for all matrices M^{-1} with a given sparsity pattern, see [2, 3, 6]. If the Frobenius norm is used, the problem can be subdivided nicely into independent sub-problems which can be treated in parallel. Unfortunately, the quality of M^{-1}, as a preconditioner for A, depends on the selected sparsity pattern. Adapting the pattern can be introduced but at the costs of a less parallel method.

In [4], incomplete block factorisation and the sparse approximate inverse technique are combined. A sub-matrix of the matrix A is selected in such a way that its approximate inverse is easily calculated. From this an approximate block factorisation of the matrix A is constructed by calculating the Schur complement S of the sub-matrix. The recursive application of this block factorisation approach provides an approximation of the inverse of the Schur complement, which is needed in the matrix-vector product with M^{-1}.

One problem of this approach is assembling the Schur complement, as it can be very expensive regarding computational costs and the resulting matrix is not as sparse as the original matrix. In order to limit the costs, small elements are dropped and/or, a certain number of non-zero elements is allowed in the sparsity pattern [8]. Unfortunately, the quality of the preconditioner M^{-1} can deteriorate dramatically, even if very small elements of S are dropped only. ARMS [11] tries to compensate this approximation error by applying an iterative solver when S^{-1} is evaluated. The incomplete factorisation of S plays the role of a preconditioner. This actually improves the convergence rate for the outer iteration. In some cases, it can make the iteration more robust, but mostly it increases the overall computing time.

In this paper we present a modification of this approach which needs one evaluation of the incomplete block factorisation in each level only. The idea is to execute a few iterative steps of GMRES on the Schur complement matrix before and after evaluating the approximate block factorisation. This procedure is similar to pre- and post-smoothing for multi-grid methods [5]. In fact, the block forward and the block backward substitution can be interpreted as restriction and prolongation, respectively. Examples show that pre- and post-smoothing reduce the number of outer iterative steps but do not reduce the overall computing time.

2 Block Factorisation

Let $A = (a_{i,j})_{i,j=1,n} \in \mathcal{R}^{n\times n}$ be a real, non-singular and sparse coefficient matrix. An incomplete (synonymously approximate) block factorisation of the matrix A is constructed in the following way: The set of *fine level* unknowns $V := \{1,\ldots,n\}$ is subdivided into two subsets F and C ($F \cap C = \emptyset$ and $F \cup C = V$), where F denotes the set of n_F unknowns that are eliminated from the matrix. The n_C unknowns in C are called the *coarse level* unknowns ($n = n_F + n_C$).

The matrix A is rearranged into the form

$$A = \begin{bmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} . \qquad (2)$$

The columns and rows of the sub-matrix $A_{FF} \in \mathcal{R}^{n_F\times n_F}$ belong to the unknowns F and the columns and rows of $A_{CC} \in \mathcal{R}^{n_C\times n_C}$ to the unknowns in C. In ILUM [9] F is selected such that A_{FF} becomes a diagonal matrix.

In this paper we consider an alternative [4]. The matrix A_{FF} is constructed to be strictly diagonal dominant, i.e.

$$\sum_{i\neq j\in F} |a_{i,j}| \leq \lambda\, |a_{i,i}| \qquad (3)$$

for all $i \in F$, where $0 \leq \lambda \leq 1$ is a given threshold (typically $\lambda = 0.3$). An algorithm for the selection of F meeting condition (3) can be obtained by modifying the classical greedy algorithm [4], which is used to find maximal independent sets in a graph.

As A_{FF} is constructed to be strictly diagonal dominant, a sparse approximate inverse Y_{FF} of A_{FF} can be computed easily, e.g. by using a Newton-type iteration [7, 4]. The (approximate) Schur complement S is defined by

$$S := A_{CC} - A_{CF} Y_{FF} A_{FC} \,. \tag{4}$$

The Schur complement S is used to calculate an approximate block factorisation of the matrix A:

$$A \approx M := \begin{bmatrix} Y_{FF}^{-1} & A_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} = \begin{bmatrix} I & 0 \\ A_{CF} Y_{FF} & I \end{bmatrix} \begin{bmatrix} Y_{FF}^{-1} & A_{FC} \\ 0 & S \end{bmatrix} . \tag{5}$$

The evaluation of M^{-1} requires the matrix-vector products with Y_{FF}, A_{FC} and A_{CF} as well as the (approximate) evaluation of S^{-1}.

If an iterative solver is applied to evaluate S^{-1}, the matrix S is not needed to be available explicitly but can be recovered in each matrix-vector product from identity (4). If the order n_C of the matrix S is large, an iterative solver will converge very slowly. Then it is more efficient to recursively apply incomplete block factorisation to S until the number of unknowns becomes reasonably small. This introduces a *multi-level* method.

For recursive factorisation the Schur complements have to be assembled explicitly. In order to limit the computational effort and the memory usage, the sparsity of S is improved by dropping small entries [8]. Thus an approximation $\widehat{S}$ of the Schur complement is used to construct an incomplete factorisation N of S.

3 Smoothing

It is an obvious idea to use a preconditioned iterative solver when evaluating S^{-1}, where the preconditioner is constructed through the approximate block factorisation N of $\widehat{S}$. Thus the iteration will be continued on a finer level, if a certain accuracy is achieved on the coarser level. This implements a *W-cycle* multi-level method [11]. In some cases W-cycles seem to be more robust, but in general they are expensive compared to V-cycles, especially on parallel computers. As a compromise, the idea of *smoothing* is introduced from multi-grid methods [5]:

If for given $q_C \in \mathcal{R}^{n_C}$ the vector $p_C = S^{-1}q_C$ is evaluated (i.e. $Sp_C = q_C$ has to be solved), ν_{pre} steps of an iterative solver are performed. This *pre-smoothing* step provides a first approximation p_C^{pre} of p_C. The incomplete factorisation N of S is applied to the new residual

$$q_C^{pre} := q_C - Sp_C^{pre} \,.$$

This provides the new approximation

$$p_C^c := p_C^{pre} + N^{-1}q_C^{pre}$$

of the sought vector $S^{-1}q_C$. The upper index c indicates an approximation of p_C corrected by a coarse level evaluation.

The approximation p_C^c is improved through ν_{post} *post-smoothing* steps by solving the linear system with coefficient matrix S and right hand side

$$q_C^{post} := q_C - S p_C^c .$$

GMRES(ν) [10] is used as smoothing scheme, where $\nu = \nu_{pre}$ is the number of pre-smoothing and $\nu = \nu_{post}$ is the number of post-smoothing steps.

In the following for all $q_C \in \mathcal{R}^{n_C}$ and for all $\nu = 0, 1, \ldots$, the vector $\mathtt{GMRES}(q_c, \nu)$ denotes the result of GMRES(ν) returned after the ν-th iterative step:

$$\mathtt{GMRES}(q_c, \nu) := \arg \min_{p_c \in K_\nu(q_c)} \| q_C - S p_C \|_2 \ , \tag{6}$$

where $\|.\|_2$ denotes the Euclidean norm in $\mathcal{R}^{n_C}$ and

$$K_\nu(q_c) := \text{span}\left\{ S^\mu q_c \right\}_{\mu=0,\ldots,\nu-1}$$

the ν dimensional Krylov space. We set $\mathtt{GMRES}(q_c, \nu) := 0$ for $\nu < 1$.

Typically the residuals in GMRES are reduced quickly in the first iterative steps only. In order to catch the fast convergence at the beginning, the iteration procedure is stopped if the residual cannot be reduced by a given factor α $(0 < \alpha < 1)$, i.e. the iteration is stopped after the ν-th step if the condition

$$\left\| q_C - S p_C^{(\nu)} \right\|_2 > \alpha \left\| q_C - S p_C^{(\nu-1)} \right\|_2 \tag{7}$$

TABLE 1: Evaluation of $p := M^{-1}q = \texttt{IBF}(0, q)$ by the forward-backward substitution scheme with l levels.

```
 1: IBF($k, q$)
 2:     if ($k = l - 1$)
 3:         solve $Sp = q$
 4:     else
 5:         $(q_F, q_C) \leftarrow q$
 6:         $q_C^{pre} \leftarrow q_C - A_{CF} Y_{FF} q_F$
 7:         $p_C^{pre} \leftarrow \texttt{GMRES}(q_C^{pre}, \nu_{pre})$
 8:         $p_C^c \leftarrow p_C^{pre} + \texttt{IBF}(k+1, q_C - S p_C^{pre})$
 9:         $p_C \leftarrow p_C^c + \texttt{GMRES}(q_C - S p_C^c, \nu_{post})$
10:         $p_F \leftarrow Y_{FF}(q_F - A_{FC} p_C)$
11:         $p \leftarrow (p_F, p_C)$
12:     endif
13: return $p$
```

holds, where $p_C^{(\mu)} := \texttt{GMRES}(q_c, \mu)$ for all $\mu = 0, 1, 2, \ldots$. As this criterion avoids unnecessary iterative steps, it is more efficient regarding the computing time in comparison to a fixed number of pre- and post-smoothing steps.

4 Forward/backward Substitution

Table 1 shows the recursive procedure evaluating $p := M^{-1}q = \mathtt{IBF}(0, q)$ for a given vector $q \in \mathcal{R}^n$. The operation $(q_F, q_C) \leftarrow q$ in step 5 selects the components q_F and q_C of the vector q which belong to the set of eliminated unknowns F and the coarse level unknowns C, respectively. The operation $p \leftarrow (p_F, p_C)$ in Step 11 is the inverse operation. On the coarsest level the linear system is solved iteratively by Step 3. The optimal number of levels has to be found by some test. It depends on the matrix A but also on the used computer architecture.

Note that for $k = l - 2$ Steps 7 and 9 can be dropped, as a solution provided by the coarsest level can not be improved by pre- or post-smoothing. Consequently a two-level method ($l = 2$) does not need any smoothing.

Obviously algorithm 1 is analogous to a multi-grid procedure, if Steps 3 and 4 are interpreted as *restriction* and Steps 10 and 11 are interpreted as *prolongation*.

5 Examples

The matrices used in the following examples are generated by discretisation of boundary value problems. The right hand sides of the linear systems are generated by random samples. Before the calculation is started, the matrix is normalised by a diagonal matrix in the following way: the entries in the main diagonal of the matrix A are non-negative and l^1-norm of each row is equal to one.

The linear systems are solved by GMRES with truncation after five residuals and restarted after 20 iterative steps. The tolerance is 10^{-4}. On the coarsest levels, the linear systems are solved with an accuracy of 10^{-3} by using GMRES with truncation after 50 residuals. Jacobi preconditioning is used.

In the following tables the columns *Outer* and *Inner* give the number of outer iterative steps and the average number of iterative steps on the coarsest level. The column *Smooth* gives the average number of matrix vector products in a single smoothing call. The columns *CPU* and *Start-up* give the computing time in seconds and the fraction of computing time spent on the factorisation. Both values were measured on a 66MHz IBM Wide Node computer.

5.1 Convection Driven Diffusion

The test case is the 3-dimensional equation for convection driven diffusion on the domain $[-1,1]^3$:

$$\begin{aligned} -\nabla\nabla u + b\nabla u &= f \quad \text{on} \quad \Omega\,, \\ u &= \phi \quad \text{on} \quad \partial\Omega\,. \end{aligned} \tag{8}$$

Table 2: Example 5.1: timings for a fixed number of smoothing steps ($n = 61828$).

	$\nu_{pre} = \nu,\ \nu_{post} = 0$			$\nu_{pre} = 0,\ \nu_{post} = \nu$			$\nu_{pre} = \nu_{post} = \nu$		
ν	*Outer*	*Inner*	*CPU*	*Outer*	*Inner*	*CPU*	*Outer*	*Inner*	*CPU*
-	90	4	109.6	90	4	109.6	90	4	109.6
1	55	4	127.1	60	4	143.1	47	4	164.5
2	47	4	145.8	47	4	151.1	37	4	189.4
3	42	4	166.4	41	4	170.5	36	4	246.8
4	34	4	170.1	33	4	171.5	34	4	301.2
5	40	4	243.9	39	4	239.8	32	4	350.0
10	32	4	392.4	32	4	393.5	26	4	601.0
15	26	5	510.5	26	4	511.8	26	5	982.3

Table 3: Example 5.1: timings for an optimised number of pre- and post-smoothing steps ($n = 61828$).

α	*Outer*	*Inner*	*Smooth*	*Start-up*	*CPU*
0.95	26	4	10.1	0.01	908.5
0.50	47	4	2.0	0.04	268.4
0.25	47	4	1.1	0.04	214.7
0.10	47	4	1.0	0.05	216.2
0.05	47	4	1.0	0.05	216.6
-	90	4	0.0	0.10	109.6

The vector-valued coefficient function b which represents a velocity field b is defined by

$$b = \left(\frac{\partial\psi}{\partial x_2} - \frac{\partial\psi}{\partial x_3}, \frac{\partial\psi}{\partial x_3} - \frac{\partial\psi}{\partial x_1}, \frac{\partial\psi}{\partial x_1} - \frac{\partial\psi}{\partial x_2}\right) \qquad (9)$$

with stream function

$$\psi = R_y(x_1^2 - 1)(x_2^2 - 1)(x_3^2 - 1)\sqrt{x_1^2 + x_2^2 + x_3^2}\,. \qquad (10)$$

The equation is discretised by the finite difference method of order 2 using a rectangular $52 \times 29 \times 41$ grid. An up-wind scheme is used for the convective term $b\nabla u$. We set $R_y = 10^5$ so that the convection term becomes dominant. This produces a highly non-symmetric matrix. The problem is difficult to be solved by an iterative solver, but can be preconditioned rather easily by incomplete factorisation methods.

Six levels are used. In the Schur complements the 6th largest entries per row and the main diagonal entry are considered only. The approximate inverse Y_{FF} is constructed by taking the inverse of the main diagonal part of Y_{FF}. The timings for a fixed number of pre- and post-smoothing steps are shown in Table 2. As expected, the number of outer iterations is reduced in all three cases, namely only post-smoothing, only pre-smoothing or the combination of pre- and post-smoothing. Unfortunately the matrix-vector products needed for the GMRES smoothing are rather expensive and therefore the overall computing time is larger than without any smoothing. Table 3 shows the timings for different values of α in the stopping criterion (7). The value $\alpha \le 0.5$, which leads to one pre- and one post-smoothing step in average, gives the best timings.

Table 4: Example 5.2: timings for a fixed number of smoothing steps ($n = 34201$).

	$\nu_{pre} = \nu$, $\nu_{post} = 0$			$\nu_{pre} = 0$, $\nu_{post} = \nu$			$\nu_{pre} = \nu_{post} = \nu$		
ν	*Outer*	*Inner*	*CPU*	*Outer*	*Inner*	*CPU*	*Outer*	*Inner*	*CPU*
-	57	12	124.5	57	12	124.5	57	12	124.5
1	51	12	190.0	30	12	151.1	29	12	175.7
2	48	12	225.4	40	12	240.3	37	12	274.3
3	31	12	198.2	34	12	212.9	35	12	333.3
4	46	11	310.6	29	12	222.2	33	11	390.0
5	46	11	367.1	35	12	293.5	43	11	585.9
10	46	12	687.4	31	12	487.1	37	12	1034.8
15	34	12	894.1	32	12	786.0	31	12	1445.5

Table 5: Example 5.2: timings for an optimised number of post-smoothing steps; no pre–smoothing ($n = 34201$).

α	*Outer*	*Inner*	*Smooth*	*Start-up*	*CPU*
0.95	49	12	8.1	0.14	427.6
0.50	30	12	1.0	0.38	152.7
0.25	30	12	1.0	0.38	153.0
0.10	30	12	1.0	0.38	152.0
0.05	30	12	1.0	0.38	152.8
-	57	12	0.0	0.46	124.5

5.2 L-Shaped Domain

The second test problem is the following variational problem: find $u : \Omega \rightarrow \mathcal{R}$ with $u|_\Gamma = b$ and

$$\int_\Omega k\nabla u \cdot \nabla v + (r\nabla u - f)\, v\, dx = 0$$

for all test functions $v : \Omega \rightarrow \mathcal{R}$ with $v|_\Gamma = 0$. The domain Ω is the 2-dimensional L-shaped set $[0,2]^2 \setminus [0,1]^2$ and Γ is the boundary portion $\{(x_1,x_2) \mid x_1 = 0 \text{ or } x_2 = 0\} \cap \partial\Omega$. We set $r^T = \frac{1}{100}(1,1)$. The coefficient function k is defined with jumps from 1 to 10^5. This makes the equation difficult to solve for iterative solvers. The variational problem is discretised by the finite element method using 11250 quadrilateral elements of order two. The drop tolerance in the Schur complement is 10^{-3}. The approximate inverse Y_{FF} is calculated with an accuracy 10^{-3} using Newton's method.

Table 4 shows the timings for fixed numbers of pre- and post-smoothing steps. The first row shows the timings without smoothing. It is sufficient to perform only a few post-smoothing steps in order to get the smallest number of outer iterative steps. This does not reduce the overall calculation time compared to a calculation without smoothing. Pre-smoothing is not as effective as post-smoothing. Table 5 presents the timings, when applying stopping criterion (7) in the post-smoothing procedure (no pre-smoothing). The last row shows the timing without smoothing. Obviously the stopping criterion has no effect on the timings if $\alpha \leq 0.5$. Only one post-smoothing step is sufficient to minimize the number of outer iterative steps.

6 Conclusion

The test examples indicate that pre- and post-smoothing by GMRES reduce the number of outer iterations significantly. Post-smoothing is preferred where one GMRES step is sufficient. In this situation the smoothing performs a simple length scaling of the vector p_C^c. The scaling is an attempt to compensate for the approximations made in the incomplete factorization of the Schur complement. A similar technique is used to smooth oscillations of the residuals in CG iterations [13]. The tests do not indicate that pre- or post-smoothing improves the robustness of the preconditioned iteration scheme.

Although the number of outer iterations is significantly reduced, length scaling does not improve the total computing time. The reason for this is the fact, that additional matrix-vector products are needed. Nevertheless, length scaling is still an interesting approach on vector and parallel computers. On these architectures the data movements in the restriction and prolongation operations are very expensive compared to floating point operations. Thus additional matrix-vector products which help to reduce the number of forward/backward substitutions can reduce the overall computing time.

References

[1] O. Axelsson and M. Neytcheva. Algebraic multilevel iteration method for Stieltjes matrices. *Num. Lin. Alg. Appl.*, 1:213–236, 1994. C655

[2] M. Benzi and M. Tuma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19(3):968–994, 1998. C655

[3] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse–sparse iteration. *SIAM J. Sci. Comput.*, 19(3):995–1023, 1998. C655

[4] L. Grosz. Preconditioning by incomplete block elimination. *J. Num. Lin. Alg. with Appl.*, 2000. to appear. C655, C657, C658, C658

[5] W. Hackbusch. *Multigrid Methods and Applications*. Springer-Verlag, Berlin, Heidelberg, New York, 1985. C656, C659

[6] T. Huckle and M. Grote. A new approach to parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comp.*, 18:838–853, 1997. C655

[7] V. Pan and R. Schreiber. An improved Newton iteration for the generalized inverse of a matrix, with applications. *SIAM J. Sci. stat. Comput.*, 12(5):1109–1130, 1991. C658

[8] Y. Saad. ILUT: A dual threshold incomplete LU factorization. *J. Num. Lin. Alg. with Appl.*, 1(4):387–402, 1994. also UMSI 92/38. C656, C659

[9] Y. Saad. ILUM: A multi-elimination ILU preconditioner for general sparse matrices. *SIAM J. Sci. Comput.*, 17:830–847, 1996. C655, C657

[10] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 20 Park Plaza, Boston, MA 02116, USA, 1996. C660

[11] Y. Saad and B. Suchomel. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. Research Report UMSI 99-107, University of Minnesota, Supercomputer Institute, 1200 Washington Avenue South, Minneapolis, Minnesota 55415, USA, 1999. C656, C659

[12] Y. Saad and J. Zhang. BILUM: Block versions of multielimination and multilevel ilu preconditioner for general sparse matrices. *SIAM J. Sci. Comput.*, 20(6):2103–2121, 1999. C655

[13] R. Weiss. *Parameter-Free Iterative Linear Solvers*. Mathematical Research, vol. 97. Akademie Verlag, Berlin, 1996. C655, C668