Multi-Level Preconditioning Techniques for General Sparse Linear Systems *

Yousef Saad[†] and Jun Zhang[‡]
Department of Computer Science and Engineering,
Minnesota Supercomputer Institute, University of Minnesota,
4-192 EE/CS Building, 200 Union Street S.E., Minneapolis, MN 55455

March 8, 1998

Abstract

We introduce multi-level block incomplete LU (BILUM) factorization preconditioning techniques for solving general sparse linear systems. The preconditioner has a multi-level structure and exhibits certain properties that are typical for multigrid method. The BILUM preconditioned Krylov subspace solver is tested for some convection-diffusion problems to show convergence that is near Reynolds number independent and near grid independent. Efficiency and robustness are also demonstrated by solving several Harwell-Boeing matrices.

1 Introduction

We consider the problem of developing efficient preconditioners for solving general large sparse linear systems of the form $Au = b$, where A is an $n \times n$ matrix. Efficient iterative solvers for such large problems usually consist of a combination of an accelerator and a good preconditioner [7]. It is generally recognized that the key to an efficient preconditioned iterative method is a high quality preconditioner.

Recently, a technique based on ILU factorization with multi-elimination (ILUM) has been used to develop robust preconditioner [6]. This type of preconditioner has a multi-level structure and exhibits certain properties that are typical for multigrid method [9]. In this paper, we consider block variants of ILUM, which we call BILUM. Block versions of ILUM preconditioner are attractive because block ILU preconditioners usually perform better than their point counterparts. In particular, ILUM may encounter difficulties when the diagonal elements of the resulting U factor are small. The problem is not as severe in a block version because of the additional freedom in selecting blocks. Block approaches are also preferred when the matrix has natural blocks that are inherited from underlying physical problems.

Block versions of ILUM were first suggested along with ILUM in [6]. Some predetermined block techniques were used in certain type of multi-level preconditioners and tested by Botta and Wubs [1]. Our blocks are constructed dynamically for general sparse matrices.

*This work was supported in part by ARPA under grant number NIST 60NANB2D1272, in part by NSF under grant CCR-9618827, and in part by the Minnesota Supercomputer Institute.

†E-mail: saad@cs.umn.edu. URL: http://www.cs.umn.edu/~saad.

‡E-mail: jzhang@cs.umn.edu. URL: http://www.cs.umn.edu/~jzhang.

2 Multi-Level Block ILU

ILUM is built on the fact that many rows of a sparse matrix can be eliminated simultaneously at a given stage of the Gaussian elimination. A set consisting of such rows is called *an independent set.* This is a set of unknowns that are not coupled by an equation. The unknowns associated with an independent set can easily be eliminated (in parallel) and yield another sparse linear system. A block independent set can be defined similarly. Nodes between blocks are not linked, but nodes within a block may be linked. In [6], several heuristic algorithms have been suggested to find maximal point independent set. In addition, a few new heuristic algorithms are introduced in [9] to find block independent set with various properties. It has been shown that, in general, BILUM with independent set found by heuristic algorithm blocking nodes with minimal degree has the best performance.

Given an independent set ordering (a permutation matrix), we may reorder the linear system with the unknowns of the independent set listed first and perform a block factorization

$$P_j A_j P_j^T = \tilde{A}_j \sim \begin{pmatrix} D_j & F_j \\ E_j & C_j \end{pmatrix} = \begin{pmatrix} I_j & 0 \\ E_j D_j^{-1} & I_j \end{pmatrix} \times \begin{pmatrix} D_j & F_j \\ 0 & A_{j+1} \end{pmatrix} = L \times U,$$

where $A_0 = A$ and D_j is a block diagonal matrix. The reduced system A_{j+1} is the Schur complement with respect to C_j as $A_{j+1} = C_j - E_j D_j^{-1} F_j$. Since D_j is block diagonal, A_{j+1} is still a sparse matrix in general. The same process may be applied recursively a few times to the consecutive reduced systems, such as A_{j+1}, until the last reduced system is small enough and can be solved by a direct or iterative solver inexpensively. In BILUM, the reduction process is performed approximately. Standard threshold strategies are employed to control the sparsity of the L and U factors [6, 8, 9]. One reason for studying this type of preconditioners is that they have a far better parallelism than the traditional incomplete LU factorization type preconditioners. Similar preconditioners have been designed and tested to show the property of near grid-independent convergence in [1, 9].

There are two main reasons to seek block generalizations of ILUM. First, the diagonal elements in D_j may be close to zero at each reduction step. The resulting reduced systems may be poorly conditioned and this may lead to inaccurate LU factors. Using blocks instead of single elements will lead to a better control of near-singularity. Second, matrices arising from certain applications have a natural block structure, in which the blocks are small dense matrices, typically of the same size. In such cases, it is natural to use this blocking for defining the block version of ILUM.

Once constructed, the application of BILUM preconditioning consists of block forward-backward solution process and an approximate solution of the last reduced system. Permutations may be done globally or level-by-level. Detailed descriptions are contained in [8, 9].

Size of the independent set. BILUM has the advantage of yielding larger independent set. This will in general produce a smaller Schur complement and the cost of solving the last reduced system is consequently reduced.

Denote by V_S the union of all subsets $B_1, \ldots, B_m$, and V_c the complement set of vertices. We are interested in lower bounds for $|V_S|$, the total size of the independent set. Let k be the number of nodes in each block and m be the number of blocks of uniform size in the independent set S_k. We make the assumption that the independent set is maximal, and that all subsets are of size $\leq k$. Any node in the vertex cover must be coupled to at least one subset from S.

We call the peripheral degree (pdeg) of a subset B of vertices the number of vertices not belonging to B which are linked to a vertex in B. The set of these vertices is usually denoted by adj(B) and pdeg$(B) = |\text{adj}(B)|$.

The size of V_S is mainly determined by the ratio pdeg$(B)/|B|$. As the blocks become smaller these ratios become smaller. Let $S = \{B_1, \ldots, B_m\}$ and denote by γ_i the ratio

$$\gamma_i = \frac{\text{pdeg}(B_i)}{|B_i|}, \quad 1 \leq i \leq m.$$

Clearly, the complement of V_S is the set of all peripheral vertices. Therefore,

$$n - |V_S| = \sum_{i=1}^{m} \gamma_i |B_i|$$

Now defining $\gamma = \max_{1 \leq i \leq m}\{\gamma_i\}$, we obtain, $n - |V_S| = \sum_{i=1}^{m} \gamma_i |B_i| \leq \gamma \sum_{i=1}^{m} |B_i| = \gamma |V_S|$. We then get the following immediate generalization of similar (scalar) results of [6]:

Proposition 2.1 *Let $S = \{B_1, B_2, \ldots, B_m\}$ be a maximal block independent set and assume that the peripheral degree of each subset B_i is* pdeg$(B_i) \leq \gamma |B_i|$. *Then,*

$$|V_S| \geq \frac{n}{1+\gamma}.$$

As the block-size increases, γ will decrease to zero provided the algorithm chooses sets that have 'a round shape'. Typically, for large k, one can expect γ to be of the order of $k^{-1/2}$ for two-dimensional meshes. In general blocks formed in a round shape have smaller peripheral degrees and are better than those formed in a thin shape which have larger peripheral degrees.

3 Numerical Experiments

Unless otherwise indicated explicitly, all tests used 20 levels of reduction. The last reduced system was solved approximately by an FGMRES(10) algorithm preconditioned by a dual threshold ILUT(p,τ) preconditioner [5], which will be referred to as the inner iteration and was terminated when either the number of iterations exceeded 10 or the (inner iteration) residual in 2-norm was reduced by a factor of 10^2. No dropping in any LU factor or the Schur complement A_1 on the first level. Beginning from the second level, elements in the Schur complements A_j and in the U factor $E_j D_j^{-1}$ on each level were dropped when they were smaller than τ multiplied by the average of the absolute values of the row in question. There was no dropping in D_j and F_j.

All matrices were considered as general unstructured sparse matrices and the right-hand sides were generated by assuming that the exact solution is a vector of all ones. The initial guess was generated randomly. The program was stopped when the initial residual (of the outer iteration) in 2-norm was reduced by a factor of 10^7. We set an upper bound of 100 for the outer FGMRES(20) (default) or FGMRES(10) iteration. The numerical experiments were conducted on a Power-Challenge XL Silicon Graphics workstation equipped with 512 MB of main memory, two 190 MHZ R10000 processors, and 1 MB secondary cache.

A Convection-Diffusion Problem. We consider the convection-diffusion equation

$$-\Delta u - \text{Re}\left(\sin x \cos \pi y \frac{\partial u}{\partial x} - \cos \pi x \sin y \frac{\partial u}{\partial y}\right) = f(x,y), \tag{1}$$

on the unit square $(0,1)^2$. Dirichlet boundary values were removed and artificial left-hand side was used. Eq. (1) was discretized by the standard 5-point upwind finite difference scheme and the

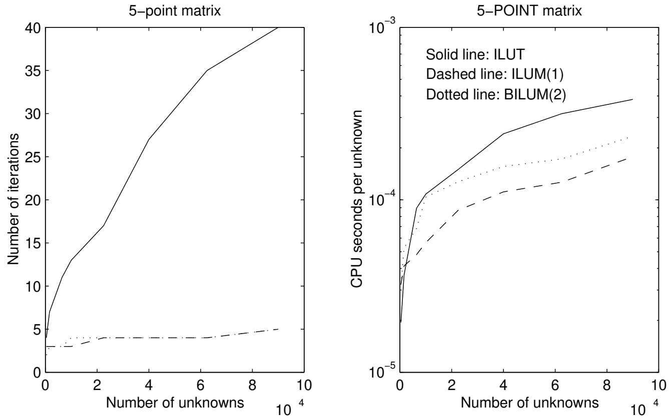

Figure 1: Comparison of different preconditioners with the 5-point matrices.

fourth-order 9-point compact scheme [3] with various (uniform) meshsize h and Re. The resulting matrices are conveniently called 5-point or 9-point matrices. In particular, we will refer to a 5-POINT matrix discretized by the 5-point upwind scheme using $\mathrm{Re} = 1$ and $h = 1/201$. This particular 5-POINT matrix has $40,000$ unknowns and $199,200$ nonzeros. Similarly, a particular 9-POINT matrix is constructed by using the 9-point compact scheme with $\mathrm{Re} = 10,000$ and $h = 1/201$. This 9-POINT matrix has $40,000$ unknowns and $357,604$ nonzeros.

We first tested our algorithms with the 5-point matrices generated by fixing $\mathrm{Re} = 1$ and varying h. The purpose of this experiment is to show how the algorithms perform as the size of the linear system increases. We compared ILUT$(10^{-4}, 20)$, ILUM(1) (1×1 block) and BILUM(2) (2×2 block). The number of iterations and the (solution) CPU time in seconds per unknown are plotted in Fig. 1.

We note from Fig. 1 that the number of ILUT iterations increased dramatically as the size of the linear system increased. On the other hand, the numbers of ILUM(1) and BILUM(2) iterations were almost independent of the size of the linear system. This suggests a near grid-independent convergence for BILUM preconditioner. The average CPU time (the CPU time in seconds per unknown) was increasing for all preconditioners when the size of the linear system increased. But the increase in ILUM-type preconditioners was moderate, especially for ILUM(1), compared with that due to ILUT. Hence, BILUM are more efficient than ILUT. (We did not have a near grid-independent CPU time because we used a fixed number of 20 level reductions.)

Next, we fixed $h = 1/201$, but varied Re for the 5-point matrices. We again compare ILUT, ILUM(1), BILUM(2) and BILUM(3) (3×3 block) with 10 levels of reduction, keeping the other parameters the same as above. The number of iterations and the solution CPU time in seconds are listed in Table 1. It is seen that ILUM(1), BILUM(2) and BILUM(3) showed convergence that is independent of the Reynolds number Re. But the convergence of ILUT varied with the magnitude of Re.

Similarly, we fixed $h = 1/81$ and varied Re to generate some 9-point matrices. We used $\tau = 10^{-4}$ and $p = 40$ in ILUT and all the coarsest level of ILUM. For ILUM-type factorizations 10 levels of reduction were used. The number of iterations and the solution CPU time in seconds are tabulated in Table 2. These results show that BILUM with larger size blocks performed better than ILUM. The ILUM preconditioned solvers showed convergence almost independent of the Reynolds number Re even when the matrix is far from diagonally dominant.

We further tested ILUT, ILUM(1) and BILUM(2) preconditioners with $\tau = 10^{-3}, p = 20$

	ILUT(10^{-4}, 20)		ILUM(1)		BILUM(2)		BILUM(3)	
Re	iter.	solu.	iter.	solu.	iter.	solu.	iter.	solu.
1	27	9.66	4	5.28	4	6.66	4	5.27
10	32	11.5	4	5.55	4	6.95	4	5.42
10^2	40	14.3	4	6.84	4	7.82	5	7.29
10^3	20	6.65	4	4.99	4	5.98	4	4.77
10^4	21	5.82	4	3.34	4	4.53	4	3.86
10^5	20	5.01	4	3.24	4	4.18	4	3.44
10^6	14	3.11	4	2.92	4	4.19	4	3.41

Table 1: Comparison of different preconditioners with the 5-point matrix.

	ILUT(10^{-4}, 40)		ILUM(1)		BILUM(2)		BILUM(3)	
Re	iter.	solu.	iter.	solu.	iter.	solu.	iter.	solu.
1	7	0.58	3	0.51	3	0.76	3	0.58
10	6	0.50	3	0.66	3	0.76	3	0.60
10^2	6	0.50	3	0.69	3	0.79	3	0.61
10^3	7	0.47	3	0.71	3	0.68	3	0.57
10^4	7	0.48	3	0.95	3	0.78	3	0.66
10^5	11	0.81	4	2.40	4	1.46	3	0.91
10^6	14	1.02	5	3.70	4	1.72	5	1.70

Table 2: Comparison of different preconditioners with the 9-point matrices.

using the 9-POINT matrix. Fig. 2 depicts the 2-norm residual reduction rate with respect to the number of iterations and the solution CPU time in seconds. In terms of the number of iterations, ILUM(1) and BILUM(2) were again much better than ILUT, with BILUM(2) being the best. In terms of solution time, both ILUM and BILUM were faster than ILUT.

Fig. 3 shows a more dramatic comparison with the Harwell-Boeing matrix SAYLR4 [2]. In this case, ILUM(1) and BILUM(2) were much more faster than ILUT both in terms of iteration counts and CPU times. The performance of the two ILUM algorithms were comparable and BILUM(2) took only slightly fewer iterations than ILUM(1) did.

Harwell-Boeing Matrices. We compare the performance of different BILUM preconditioners with 3 Harwell-Boeing matrices [2], and the 5-POINT and 9-POINT matrices. We used $\tau = 10^{-3}$ and $p = 10$ for the reduction process and the last reduced system. FGMRES(10) was used for the outer iterations. The results are given in Table 3.

For ORSREG1 and PORES2, ILUM(1) seems to be slightly faster. They performed comparably with 5-POINT. For the other matrices BILUM is more efficient. The efficiency usually increased as the size of the blocks became larger. The preprocessing cost is of the same magnitude of the solution cost, except for the two largest problems (5-POINT and 9-POINT matrices).

4 Concluding Remarks

Numerical experiments indicate that the multi-level block ILU preconditioning technique presented here is generally more robust, and often more efficient, than both ILUM and single-level

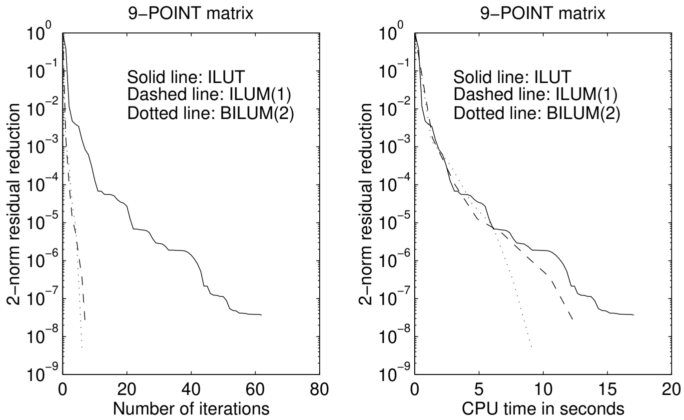

Figure 2: Comparison of different preconditioners with the 9-POINT matrix.

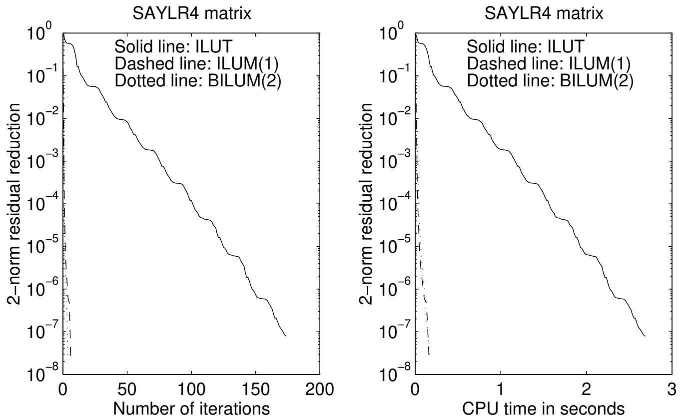

Figure 3: Comparison of different preconditioners with the SAYLR4 matrix.

ILUT preconditioning techniques. Our tests with some convection-diffusion problems showed convergence that is nearly independent of the Reynolds number and nearly independent of the meshsize of the underlying system. The results compared favorably with similar problems tested with an accelerated standard multigrid method [10]. In addition, the use of larger blocks and the related sparsity strategies are discussed in [8].

BILUM preconditioners can be viewed as multi-level variants of ILU techniques which combine the advantages of generality and robustness of the ILU technique with those of grid-independent convergence of multigrid method. The multi-level approach provides a general framework that encapsulates the coarse-grain domain decomposition-type approach as well as the fine-grain ILUM approach.

References

[1] E. F. F. Botta and F. W. Wubs, MRILU: an effective algebraic multi-level ILU-preconditioner for general sparse matrices, *SIAM J. Matrix Anal.* (to appear).

[2] I. S. Duff, R. G. Grimes, and J. G. Lewis, Sparse matrix test problems, *ACM Trans. Math. Software* **15**, 1–14 (1989).

		ILUM(1)			BILUM(2)			BILUM(3)		
Matrices	lev.	iter.	prepro.	solu.	iter.	prepro.	solu.	iter.	prepro.	solu.
ORSREG1	2	4	0.156	0.084	4	0.322	0.123	4	0.294	0.150
($n = 2,205$)	8	3	0.182	0.042	4	0.413	0.085	4	0.381	0.057
($nz = 14,133$)	14	3	0.216	0.042	4	0.445	0.089	4	0.402	0.055
	20	3	0.232	0.041	4	0.460	0.089	4	0.407	0.054
PORES2	2	–	–	–	–	–	–	–	–	–
($n = 1,224$)	8	40	0.119	0.398	27	0.179	0.339	65	0.150	0.467
($nz = 9,613$)	14	19	0.144	0.152	18	0.205	0.204	58	0.162	0.352
	20	19	0.160	0.143	20	0.223	0.229	59	0.165	0.355
ORSIRR1	2	5	0.076	0.057	5	0.121	0.072	5	0.138	0.068
($n = 1,030$)	8	7	0.080	0.044	5	0.139	0.042	5	0.133	0.033
($nz = 6,858$)	14	7	0.090	0.034	5	0.150	0.044	5	0.139	0.031
	20	7	0.092	0.034	5	0.154	0.046	5	0.147	0.030
5-POINT	2	5	2.781	5.344	5	47.94	6.717	5	35.39	6.236
($n = 40,000$)	8	6	12.03	6.008	6	66.63	7.699	6	53.24	6.421
($nz = 199,200$)	14	6	22.87	5.483	6	80.42	7.496	6	66.94	5.581
	20	6	32.59	4.843	7	91.47	8.202	6	77.43	4.694
9-POINT	2	40	5.555	81.21	–	–	–	–	–	–
($n = 40,000$)	8	10	25.07	20.73	12	68.42	21.27	6	53.05	7.648
($nz = 357,604$)	14	8	46.20	15.27	5	84.07	8.099	7	64.80	7.433
	20	8	64.70	13.84	7	95.42	10.93	6	73.11	5.588

Table 3: Comparison of different ILUM and BILUM preconditioners.

[3] M. M. Gupta, R. P. Manohar, and J. W. Stephenson, A single cell high order scheme for the convection-diffusion equation with variable coefficients, *Internat. J. Numer. Methods Fluids* **4**, 641–651 (1984).

[4] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, *SIAM J. Sci. Comput.* **14**, 461–469 (1993).

[5] Y. Saad, ILUT: a dual threshold incomplete ILU preconditioner, *Numer. Linear Algebra Appl.* **1**, 387–402 (1994).

[6] Y. Saad, ILUM: a multi-elimination ILU preconditioner for general sparse matrices, *SIAM J. Sci. Comput.* **17**, 830–847 (1996).

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

[8] Y. Saad, M. Sosonkina, and J. Zhang, Domain decomposition and multi-level type techniques for general sparse linear systems, in *Domain Decomposition Methods 10*, J. Mandel, C. Farhat and X.-C. Cai, editors, AMS, Providence, RI, 1998, pp. 200–216.

[9] Y. Saad and J. Zhang, BILUM: block versions of multi-elimination and multi-level ILU preconditioner for general sparse linear systems, Technical Report UMSI 97/126, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1997.

[10] J. Zhang, Accelerated multigrid high accuracy solution of the convection-diffusion equations with high Reynolds number, *Numer. Methods for PDEs,* **13**, 77–92 (1997).