High order ILU preconditioners for CFD problems *

Andrew Chapman † Yousef Saad ‡ Larry Wigton §

April 18, 1996

Abstract

This paper tests a number of ILU-type preconditioners for solving indefinite linear systems which arise from complex applications such as Computational Fluid Dynamics. Both point and block preconditioners are considered. The paper focuses on ILU factorization which can be computed with high accuracy by allowing liberal amounts of fill-in. A number of strategies for enhancing the stability of the factorizations are examined.

KEY WORDS Preconditioning Incomplete LU Block LU
Dropping stategies Krylov subspace methods

1 Introduction

Direct methods for solving large linear systems which result from the discretization of fluid flow problems often have prohibitive memory requirements. Although it is commonly accepted that iterative methods are required for such cases, it is also known that these methods are not as robust as direct solvers. A middle-ground strategy used to improve the robustness of iterative solvers is to precondition the original linear system with an accurate preconditioner. A preconditioning operation consists of some auxiliary process, which solves a system with A approximately. It is combined with the iterative solver to yield a faster convergence rate. The preconditioner can be itself a direct solver associated with a nearby matrix, or a few steps of an iterative technique involving A.

Preconditioning is critical in making iterative solvers practically useful. In fact, one may say that the choice of the preconditioning is far more important than that of the accelerator. When applied to common problems in Computational Fluid Dynamics (CFD), standard preconditioners are not too well understood, or even founded, theoretically. They have for

*This work was supported by NASA under Grant NAG2-904 and by NSF under grant number CCR-9214116

†Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN 55455
‡Department of Computer Science University of Minnesota, Minneapolis, MN 55455
§Boeing Commercial Airplane Group, Seattle, WA 98124

the most part been developed for the restricted class of M-matrices, but are often successfully used for indefinite problems as well. The most effective preconditioners in practice are those developed by extracting an approximate factorization. These are based on simple heuristics and the behavior of the resulting iterations are difficult to analyze. At the heart of the difficulty is the fact that often an approximation of the form

$$A \approx LU$$

is computed with some accuracy, but not much is known of the resulting matrix $M = LU$, which is used at each step of the preconditioned method. When A is ill-conditioned, as is often the case, then it is very common that M is even more ill-conditioned than A itself. In fact the forward and backward solves, i.e., the solves associated with L and U respectively, may be 'unstable' a term used to mean that the recurrences which they generate will grow exponentially. In this situation, preconditioning the system with M may have the effect of making it harder to solve rather than easier.

It is difficult to study the existence and stability of ILU factorizations for general matrices, and even more difficult to study ILU preconditioned iterative solvers for these matrices. It is not well understood for example why block techniques work typically better than point techniques. Nor is it clear whether or not increasing accuracy in the LU factorization can be useful. In fact, often more accuracy is detrimental. Diagonal compensation or perturbation can help convergence, but it is not clear whether this is due to an improvement of stability or accuracy. Preconditioners based on pattern only (level-of-fill techniques) work rather well even for indefinite problems although this is not supported by theory.

The goal of this paper is to examine a few of these facts, some of which are well-known to practitioners, with the help of experiments using matrices from CFD problems. The focus is on high order point and block ILU preconditioners as they are used in solving the systems of equations that arise from CFD problems. It is also shown that block SSOR preconditioning and deflation can be effective in some cases where ILU preconditioning fails.

2 The point preconditioners: ILU(k), ILUT, ILUD

There are a number of variations of the well-known Incomplete LU factorization scheme. These factor A such that $A = LU - E$, where E represents the matrix of fill-ins that are discarded during the factorization process. The matrices L and U are sparse lower triangular and upper triangular, respectively, and are often generated by a variant of the Gaussian Elimination process to which is added a 'dropping strategy', i.e., a rule used to discard fill-ins. The different ILU techniques differ in these dropping strategies. The corresponding subroutines are discussed briefly below. Also discussed are pivoting and block versions of the subroutines. For more details on these algorithms and their theory, see [10].

ILU(k) is a preconditioner based on a dropping strategy which uses the concept of 'level of fill' [5, 13]. When $k = 0$, the standard incomplete LU factorization with no fill-in results. In ILU(0), the factors L and U have the same pattern as the lower and upper parts of A

respectively, and each nonzero element in A is equal to the element of the product LU in the same location. The term level of fill is defined by induction, saying that an entry in the ILU factors has level of fill of $k+1$ if at least one of the parent entries in the row operations which produced it has level of fill k. Entries in the original matrix are defined to have a level of fill of zero. The preconditioner ILU(k) retains entries with level of fill up to, and including, k. The higher the value of k, the more entries retained.

ILUT uses a dual truncation dropping strategy developed in [9]. This is controlled by two parameters, a threshold drop tolerance tol, and a fill number $lfil$. All entries with magnitude less than tol multiplied by the norm of the current row are dropped, and only the largest $lfil$ entries in each row of the L and U factors are retained. It is usual when testing with this preconditioner to set tol to a fixed value, typically $tol = 10^{-4}$, and to vary $lfil$ to produce the desired fill-in.

ILUD uses threshold dropping and diagonal compensation. The threshold dropping is the same as that in ILUT, and it is controlled by a parameter tol. All entries with magnitude less than tol multiplied by the norm of the current row are dropped. These discarded fill-ins are summed up as they are dropped, and in diagonal compensation, a multiple α of this sum is added to the diagonal entry of U. This is often referred to as Modified ILU, or MILU [10].

ILUTP and ILUDP are the same as ILUT and ILUD respectively, but add threshold column pivoting. Here two columns are permuted when $|a(i,j)| * pivthresh > |a(i,i)|$, where $pivthresh$ is the pivot threshold.

Of the above methods ILU(k) is the fastest to compute the factors L and U for moderate values of k. The level of fill pattern can be obtained efficiently in a symbolic factorization routine, and after the symbolic factorization it is known how many entries there will be in the ILU factors. ILU(k) is also the simplest to use as there is only one parameter, the level of fill k. The ILUT routine may give a more accurate factorization, because when dropping entries it takes account of their magnitude. The parameter $lfil$ controls the number of entries in the ILU factors, so there is a limit to the size of the ILU factors. The dropping strategy used in ILUD is the simplest one. However, there is no way of predicting the size of the resulting ILU factors. It is not uncommon that the storage required by ILUD may be prohibitive, possibly close to that of a direct solver. No one of these preconditioners has been shown to be the best for all matrices. In practice, finding the best preconditioner for a given problem, or class of matrices associated with a problem, involves extensive testing. It is assumed here that for a given class of matrices arising from the same physical problem, the preconditioners will behave similarly.

In general the more entries in the ILU factors, the more accurate the factorization, and the better the preconditioner. This is not always the case however. A common failure of ILU factorizations is 'instability' a term which is often used to mean that the norm of $(LU)^{-1}$ can be extremely large. This is caused by long recurrences which grow exponentially in the

forward and backward triangular solutions associated with L and U. In such situations, the accelerator will generally fail on the preconditioned system, which may well have a condition number that is much worse than the un-preconditioned system. An easily calculated indicator of instability of the ILU factors is the infinity norm condition estimate $\|(LU)^{-1}e\|_\infty$, where e is a vector of all ones [3, 4].

In GMRES the residual norm is calculated as the iteration proceeds (see [11] section 3.2). Iteration is terminated if the calculated residual norm is reduced by more than a convergence tolerance ϵ. The calculation is not always correct, due to roundoff errors, and this can lead to early termination. Examples are given where this occurs.

For some matrices row and column scaling can significantly improve the performance of the iterative solver, while for others scaling leads to worse performance. It was argued in [12] that if scaling results in a deterioration of the degree of normality of the matrix, measured by the number $\|AA^T - A^TA\|/\|AA^T\|$, this will result in worse performance. Examples are given where scaling improves and hinders performance.

Another factor which affects iterative solver performance is the initial guess and the right-hand side. In this work the right-hand side is a vector with pseudo-random values between zero and one, and the initial guess is a vector of all zeros. One exception to this is the matrix BBMAT. It is supplied with a right hand side, and it is used in place of the random right hand side. A choice which often leads to easier problems is the so-called constructed right-hand side. One version of this is to calculate the right-hand side equal to the product of the matrix and a vector of all ones. The initial guess is then set to be a vector with random values between 0 and 1. Examples are given comparing random and constructed right-hand sides.

3 Block preconditioners

Matrices with block structure arise naturally in CFD problems where there is more than one unknown at each node. This structure can be exploited in *block* ILU preconditioners where Gaussian Elimination is performed in terms of blocks, i.e., small dense submatrices. For these preconditioners row operations are replaced by block row operations, and when calculating the multiple of one block row to add to another, the inverse of the diagonal block is used in place of the inverse of the diagonal entry. Block preconditioners can be faster than point preconditioners, but this depends on how the diagonal blocks are inverted, and how much speedup can be achieved for block row operations versus row operations. There are two main reasons for the interest in block preconditioners. First, there are obvious savings in storage which arise from the leaner data structure associated with block storage schemes. Second, a strong coupling exists within the equations and unknowns of an individual block, and when dropping terms from the matrix, experience shows that it is more effective to drop complete blocks, rather than individual entries from the blocks. If one considers accuracy only in the incomplete LU factorization, this may seem to be unnecessary, i.e., criteria based on magnitude may seem to be sufficient. However, block factorizations seem better behaved numerically in that they tend to be more stable.

Performance of block preconditioners can often be enhanced by perturbing the diagonal blocks slightly in order to reduce the risk of instability and improve the quality of the incomplete factors. One way of doing this is to use SVD to invert the diagonal blocks, and to vary the threshold tolerance for the singular values [2]. If B is a diagonal block to be inverted, the strategy is to replace the smallest eigenvalues or singular values by larger quantities. For example, let the Singular Value Decomposition of B be

$$B = V\Sigma U$$

where Σ is the diagonal matrix of singular values $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n$. Then before inverting B any singular value which is less than $\epsilon\sigma_1$ is replaced by $\epsilon\sigma_1$ where ϵ is a tolerance factor, see e.g. [4]. This has the effect of perturbing the matrix B by a matrix whose 2-norm is equal to $\epsilon\sigma_1$, so the norm of the perturbation is of order ϵ when measured relatively to the norm of B.

One important issue when considering block factorizations, is to define dropping strategies that are suitable for the block structure. The ILU(k) strategy is the easiest to generalize. Indeed, assuming the block-size is p the only change is that the pattern to be considered to define the level of fill for each block is the one associated with the block matrix, which is of size n/p. A block ILUT or ILUD strategy can also be defined. Instead of considering the magnitude of an individual entry to be dropped in a given block, some norm of the whole block must now be used. A whole block is dropped if its norm is small enough. In this paper the Frobenius norm is used as it is one of the simplest to compute.

A block matrix can be considered as a scalar matrix and all point preconditioners can be applied to it. However, in the common case where all zero elements within each nonzero block are considered nonzero, then the block and point versions of ILU(k) give rise to the same factorization. In this situation, it is clearly advantageous to use a block version which requires less work and memory.

In this paper the block variants of the point preconditioners described earlier will be denoted by preceding them with the letter B. Thus BILU(k) and BILUT are the block variants of ILU(k) and ILUT respectively.

4 Deflation and BSSOR preconditioning

It is shown that for certain matrices, when the size of the ILU factor infinity norm condition estimate $\|(LU)^{-1}e\|_\infty$ is large, GMRES with BSSOR preconditioning and deflation is more effective than GMRES and ILU preconditioning. BSSOR refers to a standard block extension of SSOR, see for example [10]. In deflation, the eigenvectors corresponding to the smallest eigenvalues of the preconditioned matrix are estimated as GMRES progresses, and they are added to the Krylov subspace. For further discussion of deflation see [6, 1].

5 Numerical tests

The numerical experiments are grouped by sets of matrices.[1] We start our experiments with Harwell-Boeing and FIDAP matrices using only point preconditioners. In the tests the accelerator is GMRES, with a Krylov subspace size of 50. Results are given in terms of the number of GMRES steps (ie. matrix vector multiplies) to reduce the residual norm by 10^{-8}, or if there is no convergence in 1200 steps, the reduction in residual norm in 1200 steps. Also given is the total number of non-zeros in the ILU factors, referred to as ILUnnz.

5.1 Tests with Harwell-Boeing matrices

The Harwell-Boeing matrices that are used in the tests range in size from 1080 to 5005. A brief description of each matrix is given in Table 1, where n is the dimension of the matrix, and nnz the number of non zeros.

matrix	n	nnz	description
SAYLR4	3 564	22 316	oil reservoir modeling
PORES2	1 224	9 613	oil reservoir modeling
ORSREGl	2 205	14 133	oil reservoir modeling
SHERMAN2	1 080	23 094	thermal simulation, steam injection
SHERMAN3	5 005	20 033	black oil, IMPES simulation
SHERMAN5	3 312	20 793	fully implicit black oil simulator

Table 1: Harwell-Boeing Matrices

Point versions of the five preconditioners described in section 2 were tested on these matrices. Three tests were carried out for each preconditioner. For ILU(k) these tests correspond to values of level of fill k = 0, 1, 2. For the other preconditioners, parameters were varied with the aim of producing the same fill-in as for ILU(k). The parameter *tol* in ILUT and ILUTP and the parameter α in ILUD and ILUDP were fixed at 0.0001 and 0.1. The values of other parameters are given in Table 2. For the pivoting routines ILUTP and ILUDP, the threshold pivot tolerance *pivtol* was set to 0.1.

Results of the tests are given in Table 3. The first line for each test, labeled (step)conv, gives in parenthesis the number of GMRES steps to reduce the residual norm by 10^{-8}, or without parenthesis the reduction in residual norm in 1200 GMRES steps. The second line for each test, labeled ILUnnz, gives the number of non-zeros in the incomplete L and U factors.

The performance of the preconditioners can be compared by looking at the number of iterations to converge for similar ILUnnz. It can be seen that ILUTP has worse convergence results than ILUT, and ILUDP has better convergence results than ILUD. No preconditioner is the best for all matrices, and in general they give similar results for similar ILUnnz. In

[1] All the matrices referred to in this paper are accessible via the internet. See the web pages in `http://www.cs.umn.edu/Research/arpa/SPARSKIT` for pointers.

matrix	ILUT,ILUTP *lfil*			ILUD, ILUDP *tol*		
	test 1	test 2	test 3	test 1	test 2	test 3
SAYLR4	3	5	7	0.000 25	0.000 247 5	0.000 245
PORES2	6	10	14	0.01	0.001	0.000 1
ORSREG1	3	5	7	0.002	0.000 4	0.000 08
SHERMAN2	18	20	22	0.000 5	0.000 05	0.000 005
SHERMAN3	3	5	7	0.1	0.05	0.025
SHERMAN5	7	10	13	0.01	0.003	0.000 9

Table 2: Harwell-Boeing Matrices, point preconditioner parameters

all cases convergence improves as ILUnnz increases, and except for the case of PORES2 and ILUD preconditioning, convergence is achieved for large enough ILUnnz. For the Harwell-Boeing matrices the general advice would be to use ILU(0), as it is faster than the other preconditioners, and the size of the incomplete L and U factors is known.

From Table 3 it can be noted that in the case of the matrix SAYLR4 and the preconditioner ILUD, the amount of fill-in is sensitive to the parameter *tol*. Small changes in *tol*, from 0.00025 to 0.0002475, produce large changes in ILUnnz, from 12 354 to 39 544. This is undesirable behavior, and it shows that it is difficult to achieve the desired fill-in for ILUD preconditioning by adjusting *tol*.

The infinity norm condition estimate $\|(LU)^{-1}e\|_\infty$ has values of the order of 10^4, 10, 10^{-1}, 10^4, 10^{10}, 10^1 in tests for the matrices SAYLR4, PORES2, ORSREG1, SHERMAN2, SHERMAN3, SHERMAN5 respectively. This indicates stable factorization in all tests.

Table 4 shows the effect of row then column scaling for the case of ILU(0) preconditioning. Here, the 2-norms are used, so that row-scaling divides each row of the matrix by its 2-norm and similarly, column scaling divides each column by its 2-norm. Columns 2 and 3 in the table give the number of iterations to converge without and with scaling, and columns 4 and 5 give the number $\|AA^T - A^TA\|/\|AA^T\|$, a measure of the degree of normality of the matrix. It can be seen that scaling improves performance, except for the case of the symmetric matrix SAYLR4 (with $\|AA^T - A^TA\|/\|AA^T\| = 0$). Here scaling destroys symmetry and leads to worse performance. It is also interesting to note that with the exception of SHERMAN3, this improvement seems to correlate with the degree of normality as measured above. Scaling was found to not produce any significant changes in other tests for this report, so the problems were not scaled, as this makes the results easier to reproduce.

It was mentioned in Section 2 that a random right hand side is us used in all tests, and that a constructed right hand side leads to an easier problem. This is confirmed by the results in the last column of table 3, which give the number of GMRES steps to converge for ILU(0) preconditioning and a random rhs. These are 60, 48, 62, 44, 233, 36 for the matrices SAYLR4, PORES2, ... SHERMAN5, respectively. For the same tests with a constructed right hand side, the numbers of steps to converge are 43, 39, 39, 14, 113, 32. The reason that a random right-hand side is used, is because it is thought to be more representative of real problems.

The Harwell-Boeing matrices are included in this report as a base case, to show that

matrix			ILUD	ILUT	ILUDP	ILUTP	ILU(k)
SAYLR4	test 1	(step)conv	(118)	(70)	(118)	(70)	(60)
		ILUnnz	12 354	21 177	12 354	21 172	22 316
	test 2	(step)conv	(85)	(56)	(85)	(56)	(54)
		ILUnnz	39 544	35 382	39 544	35 382	38 738
	test 3	(step)conv	(42)	(51)	(42)	(50)	(50)
		ILUnnz	72 872	49 496	72 872	49 496	63 476
PORES2	test 1	(step)conv	0.9E+0	(32)	0.9E+0	(31)	(48)
		ILUnnz	8 428	12 846	8 637	12 941	9 613
	test 2	(step)conv	0.1E+1	(15)	0.1E+1	(22)	(21)
		ILUnnz	23 107	24 060	23 918	24 043	18 953
	test 3	(step)conv	0.9E+0	(12)	(62)	(16)	(17)
		ILUnnz	51 790	33 412	47 613	33 366	33 829
ORSREG1	test 1	(step)conv	(20)	(43)	(20)	(43)	(62)
		ILUnnz	14 053	12 704	14 053	12 699	14 133
	test 2	(step)conv	(11)	(16)	(11)	(16)	(17)
		ILUnnz	21 385	20 802	21 385	20 802	24 853
	test 3	(step)conv	(7)	(11)	(7)	(11)	(16)
		ILUnnz	33 663	30 413	33 663	30 413	41 437
SHERMAN2	test 1	(step)conv	0.1E+1	(145)	0.5E+01	0.1E+1	(45)
		ILUnnz	18 881	16 303	67 488	22 310	23 094
	test 2	(step)conv	(350)	(25)	0.1E+00	(102)	(27)
		ILUnnz	24 728	32 402	63 037	38 515	42 463
	test 3	(step)conv	(76)	(12)	0.1E+00	(78)	(7)
		ILUnnz	35 990	35 389	83 702	42 286	68 336
SHERMAN3	test 1	(step)conv	(77)	(216)	(77)	(218)	(233)
		ILUnnz	19 835	18 851	19 835	18 850	20 033
	test 2	(step)conv	(36)	(96)	(36)	(96)	(149)
		ILUnnz	26 499	30 070	26 499	30 070	32 943
	test 3	(step)conv	(32)	(46)	(32)	(46)	(50)
		ILUnnz	32 039	41 124	32 039	41 124	52 157
SHERMAN5	test 1	(step)conv	(36)	(30)	(36)	(31)	(36)
		ILUnnz	29 594	21 824	29 662	21 851	20 793
	test 2	(step)conv	(25)	(24)	(25)	(24)	(24)
		ILUnnz	48 942	30 921	48 824	30 921	37 461
	test 3	(step)conv	(17)	(21)	(17)	(21)	(19)
		ILUnnz	82 744	39 229	82 744	39 229	63 943

Table 3: Harwell-Boeing Matrices, point ILU preconditioners

GMRES and point ILU preconditioning works well on these matrices, and their behavior is stable and predictable. Scaling usually improves convergence, but if it leads to the matrix becoming more non-normal, it can lead to worse performance. The right hand side and initial guess play a part in determining the performance of GMRES.

matrix	(step)		$\|AA^T - A^TA\|/\|AA^T\|$	
	without scaling	with scaling	without scaling	with scaling
SAYLR4	(60)	(84)	0	0.077
PORES2	(48)	(41)	1.4	0.65
ORSREG1	(62)	(62)	0.40	0.077
SHERMAN2	(44)	(12)	1.4	0.92
SHERMAN3	(233)	(2)	0.00000033	0.061
SHERMAN5	(36)	(31)	1.3	0.20

Table 4: Convergence and degree of normality for systems with and without scaling

5.2 Tests with the FIDAP matrices

A collection of matrices has been gathered from test problems used in the finite element package FIDAP. They are listed in Table 5. Linear systems associated with these matrices are often more difficult to solve than the Harwell-Boeing matrices. It can be noted that for their size, n=656 to 2,203, the FIDAP matrices have a relatively large number of entries per row, 24-42. The matrix ex3.mat is symmetric, and the others are non-symmetric.

matrix	n	nnz	description
ex3.mat	1 821	52 685	2D Flow Past a Cylinder in Freestream
ex20.mat	2 203	69 981	2D Attenuation of a Surface Disturbance
ex21.mat	656	19 144	2D Growth of a Drop from a Nozzle
ex27.mat	974	40 782	2D Crystal Growth Simulation

Table 5: FIDAP matrices

The tests on the FIDAP matrices are similar to those for the Harwell-Boeing matrices. Three tests were carried out for each matrix, and for ILU(k) these correspond to level of fill values k = 5, 10, 15. The parameters for the other preconditioners were adjusted to produce similar fill-in, and they are given in Table 6. The pivot tolerance for ILUTP and ILUDP was set to 0.1, and the factor α in ILUD and ILUDP was set to 1.0.

Table 7 gives results for the FIDAP matrices. The first line for each test gives the number of GMRES steps for the residual norm calculated in GMRES to reduce by more than 10^{-8}. A maximum of 1200 GMRES steps was allowed. The second line gives the true reduction in residual norm, $\|b-Ax\|/\|b-Ax_0\|$. If the residual norm calculated in GMRES is correct, this number is less than 10^{-8}, and any number larger than 10^{-8} indicates instability or roundoff

	ILUT		ILUD	ILU(k)
	tol	lfil	tol	lfil
test1	10^{-6}	40	10^{-4}	5
test2	10^{-7}	60	10^{-6}	10
test3	10^{-8}	80	10^{-8}	15

Table 6: Preconditioner parameters for FIDAP matrices

error in GMRES. The third line of each test gives the number on non-zero entries in the incomplete LU factors. The last line gives the infinity norm condition estimate $\|(LU)^{-1}e\|_\infty$. For problems where $\|(LU)^{-1}e\|_\infty$ is greater than 10^{20}, no attempt was made to solve, and this is indicated by the symbol † in the first two lines of each test.

The first thing to notice with the FIDAP tests is the high value of the level of fill required for ILU(k) to converge. Also notice that the preconditioners are in general unstable with respect to ILUnnz. For example, for matrix ex3.mat with ILUT preconditioning, moving from test 2 to test 3 there is a relatively small (14%) increase in ILUnnz from 141,285 to 160,613, and this results in a problem for which there is no convergence in 1200 iterations, converging in just 13 iterations.

The incorrect calculation of residual norm in GMRES is evident in the result for the matrix EX3.mat with ILU(15) preconditioning (test 3). GMRES terminates after 17 steps when the reduction in residual norm is only 0.5E-03. This test was repeated in real*16 arithmetic, and GMRES terminated after 20 steps with a true reduction in residual norm of 0.23×10^{-10}.

The tests show that the FIDAP matrices are examples of CFD matrices that are difficult to solve. The ILU preconditioners require a high level of fill to be effective, and preconditioner performance is unstable with respect to ILUnnz. There are also roundoff errors that lead to incorrect calculation of the residual norm in GMRES.

5.3 Tests with the BARTH matrices

The matrices BARTHT1A, BARTHT2A, BARTHS1A, and BARTHS2A were supplied by Tim Barth of NASA Ames. They are for a 2D high Reynolds number airfoil problem, with a one equation turbulence model. The 1A and 2A matrices are for distance-1 and distance-2 neighbor finite volume models respectively. The S and T matrices are for two different grids. The T matrices grid has a concentration of elements unrealistically close to the airfoil. They are more difficult to solve than the S matrices, which have a more realistic grid. The matrices have a 5x5 block structure. Two rows in each block are for the momentum equations, and one row each is for mass balance, energy balance, and turbulence model. There is a zero entry in each block, which arises from applying the mass balance as a constraint, and blocks representing boundary conditions contain several zeros. The main interest is in solving the distance-2 matrix problems with a preconditioner constructed from the distance-1 matrices. Results are also given for solving the distance-1 problem with a distance-1 preconditioner,

matrix			ILUD	ILUT	ILUDP	ILUTP	ILU(k)
ex3.mat	test1	(step)	(1201)	†	(1201)	†	(1201)
		conv	0.8E+00	†	0.9E+00	†	0.1E+01
		ILUnnz	150 489	111 846	220 657	127 767	139 219
		$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+06	0.2+221	0.8E+04	NaN	0.6E+03
	test2	(step)	(1201)	(1201)	(1201)	†	(1201)
		conv	0.8E-03	0.1E+01	0.8E+00	†	0.9E+00
		ILUnnz	160 719	141 285	187 597	139 749	179 359
		$\|\|(LU)^{-1}e\|\|_\infty$	0.4E+05	0.1E+13	0.2E+05	NaN	0.1E+03
	test3	(step)	(14)	(13)	(16)	†	(17)
		conv	0.9E-07	0.8E-07	0.4E-05	†	0.5E-03
		ILUnnz	172 985	160 613	188 287	147 557	186 699
		$\|\|(LU)^{-1}e\|\|_\infty$	0.4E+03	0.5E+03	0.4E+03	0.2+146	0.5E+03
ex20.mat	test1	(step)	(1201)	(1201)	(1201)	†	(147)
		conv	0.5E+00	0.1E+01	0.7E-04	†	0.7E-08
		ILUnnz	253 072	168 948	386 122	170 316	293 831
		$\|\|(LU)^{-1}e\|\|_\infty$	0.1E+09	0.4E+10	0.8E+10	0.5E+82	0.3E+10
	test2	(step)	(12)	(94)	(7)	†	(28)
		conv	0.2E-08	0.1E-06	0.3E-09	†	0.2E-11
		ILUnnz	327 042	242 585	442 258	252 878	341 633
		$\|\|(LU)^{-1}e\|\|_\infty$	0.3E+11	0.1E+09	0.2E+09	0.3E+21	0.6E+09
	test3	(step)	(4)	(74)	(4)	(74)	(28)
		conv	0.3E-09	0.5E-06	0.1E-09	0.9E-08	0.2E-11
		ILUnnz	337 873	290 948	463 065	322 146	341 633
		$\|\|(LU)^{-1}e\|\|_\infty$	0.3E+09	0.5E+09	0.3E+09	0.2E+10	0.6E+09
ex21.mat	test1	(step)	(1201)	(1201)	(1200)	(1201)	(24)
		conv	0.5E+00	0.1E+01	0.6E-00	0.1E+01	0.1E-07
		ILUnnz	49 235	46 379	83 054	48 619	69 030
		$\|\|(LU)^{-1}e\|\|_\infty$	0.7E+08	0.6E+09	0.9E+08	0.2E+16	0.2E+09
	test2	(step)	(21)	(161)	(12)	(145)	(36)
		conv	0.3E-08	0.3E-03	0.5E-08	0.2E-07	0.7E-08
		ILUnnz	70 611	60 605	93 244	66 941	72 648
		$\|\|(LU)^{-1}e\|\|_\infty$	0.9E+08	0.5E+09	0.3E+09	0.3E+10	0.1E+10
	test3	(step)	(6)	(16)	(5)	(18)	(36)
		conv	0.4E-09	0.3E-06	0.1E-08	0.2E-08	0.7E-08
		ILUnnz	72 208	68 697	95 472	80 370	72 648
		$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+10	0.1E+11	0.1E+10	0.1E+10	0.1E+10
ex27.mat	test1	(step)	(245)	†	(16)	(583)	(247)
		conv	0.7E-08	†	0.9E-08	0.4E-07	0.4E-01
		ILUnnz	71 987	68 070	94 971	66 823	136 842
		$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+08	0.5E+26	0.2E+07	0.4E+11	0.8E+11
	test2	(step)	(23)	(88)	(4)	(14)	(26)
		conv	0.4E-08	0.4E-06	0.1E-08	0.5E-08	0.3E-09
		ILUnnz	107 212	97 865	126 952	95 856	139 863
		$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+07	0.8E+08	0.2E+07	0.3E+07	0.5E+08
	test3	(step)	(12)	(65)	(3)	(8)	(26)
		conv	0.4E-09	0.1E-06	0.6E-11	0.1E-08	0.3E-09
		ILUnnz	126 787	117 581	142 885	121 472	139 863
		$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+07	0.6E+07	0.2E+07	0.2E+07	0.5E+08

Table 7: FIDAP matrices, point ILU preconditioners

and the distance-2 problem with a distance-2 preconditioner. Table 8 gives the dimension of the matrices (n), the number of non-zeros (nnz), and the number of non-zero blocks ($nnzb$). Figure 1 shows the non-zero patterns of the BARTHT matrices. The patterns for the BARTHS matrices are similar.

matrix	n	nnz	nnzb	description
BARTHT1A	14 075	439 628	19 245	SMALL AEROFOIL 2D N-S WITH TURB. HIGH RE, distance 1
BARTHS1A	15 735	498 620	21 569	SMALL AEROFOIL 2D N-S WITH TURB. HIGH RE, distance 1
BARTHT2A	14 075	955 868	52 469	SMALL AEROFOIL 2D N-S WITH TURB. HIGH RE, distance 2
BARTHS2A	15 735	1 105 164	60 413	SMALL AEROFOIL 2D N-S WITH TURB. HIGH RE, distance 2

Table 8: Barth matrices

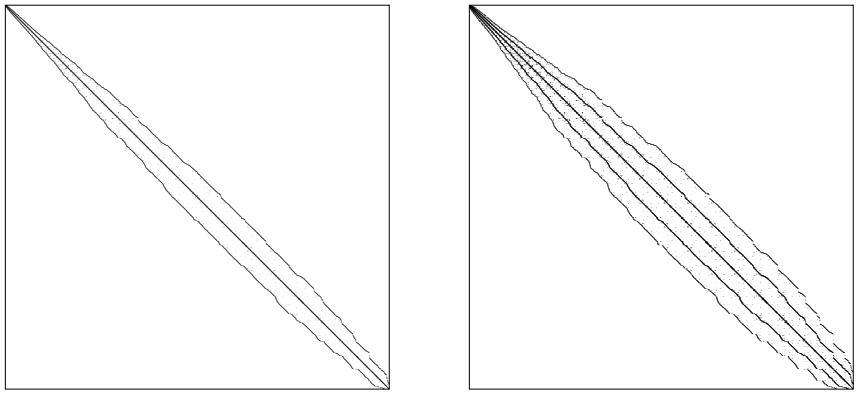


Figure 1: Non-zero pattern for Barth T matrices

Table 9 gives results for ILU(k) preconditioning with the preconditioner constructed from the distance-1 matrix. Columns 2, 3, and 4 are for regular ILU(k) preconditioning. Columns 5, 6, and 7 are for the case where the zero entries in blocks are counted as part of the non-zero pattern. This is refered to as padding. The infinity norm condition estimate $\|(LU)^{-1}e\|_\infty$ is less than 10^9 for all the tests.

Table 10 gives results for solving the same problem using ILUT preconditioning and only considering the easiest matrix BARTHS1A. Tables 11 and 12 are for the block preconditioners BILU(k) and BILUT. Note that the ILUnnzb in tables 11 and 12 is the number of non-zero blocks in the ILU factors, and not the number of entries. If the ILUnnzb numbers in Table

matrix		no padding of blocks			padded blocks		
		k=0	k=1	k=2	k=0	k=1	k=2
BARTHT1A	(step)conv	(95)	(43)	(32)	(95)	(43)	(32)
	ILUnnz	439 628	628 460	921 404	481 125	659 825	958 775
BARTHS1A	(step)conv	0.3E-1	(49)	(36)	0.3E-1	(49)	(36)
	ILUnnz	498 620	732 097	1 073 471	539 225	761 925	1 111 325
BARTHT2A	(step)conv	(450)	(194)	(123)	(556)	(194)	(123)
	ILUnnz	439 628	628 460	921 404	481 125	659 825	958 775
BARTHS2A	(step)conv	0.9E-1	(185)	(128)	0.8E-1	(183)	(129)
	ILUnnz	498 620	732 097	1 073 471	539 225	761 925	1 111 325

Table 9: ILU(k) preconditioner constructed from distance-1 matrix

matrix		lfil=10	lfil=20	lfil=30	lfil=40
BARTHT1A	conv	0.1E+1	0.1E+1	0.1E+1	0.1E+1
	ILUnnz	268 846	536 923	804 695	1 072 450
	$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+14	0.2E+15	0.8E+14	0.7E+11

Table 10: ILUT preconditioner constructed from distance-1 matrix

11 are multiplied by 25 the result is the same as the ILUnnz numbers in the last 3 columns of Table 9.

matrix		k=0	k=1	k=2
BARTHT1A	(step)conv	(94)	(42)	(31)
	ILUnnzb	19 245	26 393	38 351
BARTHS1A	(step)conv	0.3E-1	(48)	(35)
	ILUnnzb	21 569	30 477	44 453
BARTHT2A	(step)conv	(545)	(190)	(120)
	ILUnnzb	19 245	26 393	38 351
BARTHS2A	(step)conv	0.7E-1	(176)	(121)
	ILUnnzb	21 569	30 477	44 453

Table 11: BILU(k) preconditioner constructed from distance-1 matrix

To understand the results in tables 9 10 11 and 12 above, consider first the level of fill routines ILU(k) and BILU(k). The 5x5 Barth matrix blocks contain zero entries which result from applying the compressibility condition, or boundary conditions. ILU(k) gives similar results whether or not the blocks are padded. ILU(k) with padded blocks gives virtually identical results to BILU(k). This is to be expected, as the underlying preconditioners are the same in exact arithmetic. For the threshold preconditioners, the block version BILUT gives better results than the point version ILUT. Level of fill preconditioners for both point and block implementations perform better than threshold preconditioners.

matrix		lfil=0	lfil=1	lfil=2	lfil=3	lfil=4	lfil=5
BARTHT1A	(step)conv	1.0E-0	(895)	(291)	(139)	(90)	(70)
	ILUnnzb	2 815	5 629	11 132	16 460	21 659	26 995
BARTHS1A	(step)conv	1.0E+0	0.9E+0	0.8E-1	(290)	(195)	(141)
	ILUnnzb	3 147	6 286	12 434	18 312	23 999	29 946
BARTHT2A	(step)conv	1.0E+0	1.0E+0	1.0E+0	(928)	(473)	(371)
	ILUnnzb	2 815	5 629	11 132	16 460	21 659	26 995
BARTHS2A	(step)conv	1.0E+0	1.0E+0	1.0E+0	1.0E+0	(597)	(337)
	ILUnnzb	3 147	6 286	12 434	18 312	23 999	29 946

Table 12: BILUT preconditioner constructed from distance-1 matrix

Tables 13 and 14 give results for BILU(k) and BILUT preconditioners constructed from the distance-2 matrices. For similar fill-in, BILU(k) works better than BILUT. Figure 1 shows the non-zero pattern for BILU(0), and figure 2 for BILU(10), which has similar ILUnnzb. Comparing the patterns shows that BILUT(10) does not have much of the outer band which corresponds to distance 2 interaction in the finite volume formulation, and it has some vertical lines in the pattern. The matrix was column scaled, row scaled, and row and column scaled but this did not change the non-zero pattern or the convergence results.

To try to resolve the problem, another drop scheme was tried. This used the same dual truncation strategy as BILUT, but keeps all blocks in the non-zero pattern of the original matrix. Results are given in Table 15. The new drop strategy gives a fill-in pattern similar to that for BILU(k), and the convergence is also similar.

It can be concluded that the threshold dropping method is not suitable for the BARTH matrices, and that it is beneficial to keep all entries in the L and U factors which correspond to the ILU(0) pattern. An observation is that as the number of entries or blocks per row in the ILU factors gets larger, the more the drop strategy has to decide what to keep and what to drop, and the better the level of fill strategy compared to the threshold strategy. Thus for the level 1 matrices, with an average of 31 entries or 6 blocks per row, the point preconditioner ILU(k) is clearly better than ILUT, and there is less advantage in using the block preconditioner BILU(k) compared to BILUT. For the level 2 matrices, with an average of 20 blocks per row, the advantage of the block preconditioner BILU(k) over BILUT is clear. As an aside, note that Table 14 gives examples where increasing ILUnnzb does not produce better convergence.

matrix		k=0	k=1	k=2
BARTHT2A	(step)	(62)	(33)	(23)
	ILUnnzb	52 469	94 277	152 429
BARTHS2A	(step)	(73)	(32)	(23)
	ILUnnzb	60 413	110 603	175 203

Table 13: BILU(k) preconditioner constructed from distance-2 matrix

matrix		lfil=0	lfil=2	lfil=4	lfil=6	lfil=8
BARTHT2A	(step)conv	1.0E+0	1.0E+0	(661)	(741)	(537)
	ILUnnzb	2 815	11 185	22 082	32 921	43 620
	$\|\|(LU)^{-1}e\|\|_\infty$	0.7E+6	0.6E+7	0.8E+8	0.7E+8	0.2E+9
BARTHS2A	(step)conv	1.0E+0	1.0E+0	1.0E+0	0.9E+0	1.0E+0
	ILUnnzb	3 147	12 460	24 495	36 443	48 280
	$\|\|(LU)^{-1}e\|\|_\infty$	0.1E+7	0.3E+8	0.2E+9	0.6E+9	0.2E+10

matrix		lfil=10	lfil=12	lfil=14	lfil=16	lfil=18
BARTHT2A	(step)conv	(489)	(681)	0.1E+0	0.1E+0	0.1E+0
	ILUnnzb	54 218	64 600	74 983	85 298	95 695
	$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+9	0.2E+9	0.2E+9	0.3E+9	0.2E+9
BARTHS2A	(step)conv	1.0E+0	1.0E+0	1.0E+0	1.0E+0	1.0E+0
	ILUnnzb	60 078	71 769	83 585	95 261	106 880
	$\|\|(LU)^{-1}e\|\|_\infty$	0.2E+10	0.5E+10	0.1E+11	0.3E+11	0.7E+11

Table 14: BILUT preconditioner constructed from distance-2 matrix.

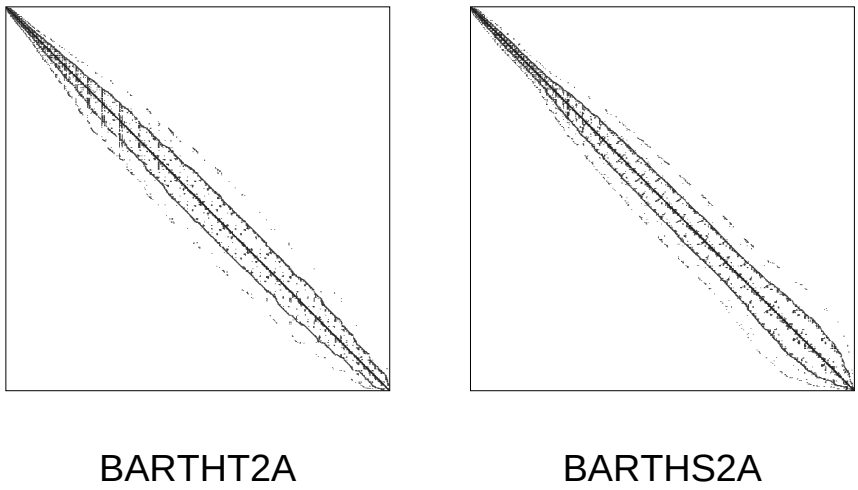


Figure 2: Non-zero pattern for BILUT(10) factors

matrix		lfil=0	lfil=2	lfil=4	lfil=6	lfil=8
BARTHT2A	(step)	(62)	(37)	(35)	(35)	(33)
	ILUnnzb	52 469	63 277	73 923	84 371	94 611
BARTHS2A	(step)	(73)	(34)	(30)	(30)	(28)
	ILUnnzb	60 413	72 320	84 114	95 826	107 462

Table 15: BILUT preconditioner with modified drop strategy constructed from the distance-2 matrix

5.4 Tests with the SIMON matrices

The matrix WIGTO966 is from an Euler equation model. The RAEFSKY, VENKAT, and BBMAT matrices were supplied by Horst Simon [2]. These are all CFD matrices. A brief description is given for each matrix in Table 16. The columns n, nnz, and blk are for dimension, number of non zero entries, and block size.

matrix	n	nnz	blk	description
RAEFSKYl	3 242	294 276	-	INCOMPRESSIBLE FLOW IN PRESSURE DRIVEN PIPE
RAEFSKY3	21 200	1 488 768	8	FLUID STRUCTURE INTERACTION TURBULENCE PROBLEM
RAEFSKY5	6 316	168 658	6	LANDING HYDROFOIL AIRPLANE FSE MODEL
VENKAT01	62 424	1 717 792	4	UNSTRUCTURED 2D EULER SOLVER, V. VENKATAKRISHNAN NASA TIME STEP = 0
VENKAT25	62 424	1 717 792	4	UNSTRUCTURED 2D EULER SOLVER, V. VENKATAKRISHNAN NASA TIME STEP = 25
VENKAT50	62 424	1 717 792	4	UNSTRUCTURED 2D EULER SOLVER, V. VENKATAKRISHNAN NASA TIME STEP = 50
WIGTO966	3 864	238 252	4	EULER EQUATION MODEL
BBMAT	38 744	1 771 722	4	BEAM + BAILEY 2D AIRFOIL EXACT JACOBIAN

Table 16: Simon Matrices

Table 17 has results for solving the matrices using ILU subroutines. The tests labeled 1, 2, and 3 in the table are for level of fill in ILU(k) is equal to 0, 1, 2, and the parameters for the other preconditioners were adjusted to produce similar ILUnnzb. The RAEFSKY and VENKAT matrices all converge with ILU(0) preconditioning, and they will be discussed no further. For the matrices WIGTO966 and BBMAT, the condition number estimate $\|(LU)^{-1}e\|_\infty$ is high, and the next paragraphs discuss techniques to produce factorizations with lower condition number.

One method used in ILU preconditioning to lower the condition number estimate $\|(LU)^{-1}e\|_\infty$ is to perturb the diagonal blocks. Table 18 gives results for BILU(k) preconditioning, with SVD used to invert and perturb diagonal blocks (as shown in Section 3). The value of the SVD threshold is the value by which the diagonal is perturbed, and the column for SVD threshold equal to 10^{-14} can be considered as regular BILU(k).The column for SVD threshold of 10^{-1} is for large perturbation. In all tests as the perturbation increases, the condition number estimate $\|(LU)^{-1}e\|_\infty$ decreases. For BARTHT1A and BARTHT2A this hinders convergence. For WIGTO966 it improves convergence. For BBMAT the factorization becomes more stable, indicated by $\|(LU)^{-1}e\|_\infty$ decreasing, but this does not lead

[2] These matrices are available through the WWW. See the home pages of the second author at: `http://www.cs.umn.edu`.

to convergence. Note that for the matrix BBMAT with regular BILU(k) preconditioning (SVD threshold = 10^{-14}), the condition number estimate $\|(LU)^{-1}e\|_\infty = 0.2 \times 10^{14}$ in table 18 is much lower than the condition number estimate $\|(LU)^{-1}e\|_\infty = 0.3 \times 10^{50}$ in table 17 for the same matrix but with ILU(k) preconditioning. This is because the 4x4 blocks in BBMAT contain a large number of zero entries which are in the BILU(0) pattern, but not in the ILU(0) pattern. Multiplying ILUnnzb in Table 18 by 16 gives ILUnnz = 3 344 096 for BILU(0) as compared to 1 771 722 for ILU(0). This can be contrasted with the BARTH matrices, where blocks contain fewer zeros, as can be seen when comparing ILUnnz = 498 620 or 539 225 in Table 9 for ILU(0) with and without padded blocks.

matrix			ILUD	ILUT	ILUDP	ILUTP	ILU(k)
RAEFSKYl	test 1	(step)conv	(49)	(18)	(40)	(19)	(36)
		ILUnnz	200 940	313 912	190 317	315 977	293 409
RAEFSKY5	test 1	(step)conv	(18)	(5)	(19)	(5)	(5)
		ILUnnz	153 172	166 398	153 337	172 637	167 178
RAEFSKY3	test 1	(step)conv	†	(36)	†	(36)	(186)
		ILUnnz	1 270 282	1 287 349	21 199	1 313 105	1 488 768
VENKAT0l	test 1	(step)conv	†	(20)	†	(20)	(21)
		ILUnnz	1 719 187	1 867 216	1 646 406	1 867 216	1 717 792
VENKAT25	test 1	(step)conv	†	(256)	†	(253)	(290)
		ILUnnz	1 711 638	1 867 236	2 038 123	1 867 251	1 717 763
VENKAT50	test 1	(step)conv	†	(348)	†	(350)	(445)
		ILUnnz	1 514 585	1 867 274	1 760 756	1 867 277	1 717 777
WIGTO966	test 1	(step)conv	0.1E+1	†	0.1E+01	0.1E+1	0.1E+1
		ILUnnz	230 525	223 400	238 111	224 673	238 252
		$\|(LU)^{-1}e\|_\infty$	0.18E+16	0.11E+47	0.6E+19	0.60E+09	0.22E+18
	test 2	(step)conv	0.1E+1	†	0.1E+01	(783)	0.1E+1
		ILUnnz	434 205	443 407	526 883	445 614	418 661
		$\|(LU)^{-1}e\|_\infty$	0.11E+17	0.52E+28	0.5E+11	0.23E+08	0.25E+19
	test 3	(step)conv	0.1E+1	0.1E+1	0.1E+01	(200)	0.1E+1
		ILUnnz	639 513	659 983	670 239	662 823	631 759
		$\|(LU)^{-1}e\|_\infty$	0.2E+10	0.6E+18	0.4E+06	0.2E+06	0.5E+11
BBMAT	test 1	(step)conv	0.10E+01	†	†	†	†
		ILUnnz	1 696 203	1 907 963	1 909 688	1 932 579	1 771 722
		$\|(LU)^{-1}e\|_\infty$	0.1E+17	0.4E+106	0.1D+94	0.3E+390	0.3E+50

Table 17: SIMON matrices, point ILU preconditioners

Another change which can be made to ILU preconditioners to reduce $\|(LU)^{-1}e\|_\infty$ is to drop all entries in the original matrix which are outside of a band about the diagonal. This is referred to as banded ILU. A similar change is to drop all entries which are outside of diagonal blocks, and this gives rise to a block diagonal preconditioning. The diagonal blocks decouple and for small block size it is efficient to invert them using dense matrix techniques like LU decomposition. BSSOR preconditioning uses the same inverted diagonal blocks, but it also uses information from off-diagonal blocks. The above preconditioners have in common the use of more information from matrix blocks close to the diagonal, and less

matrix		SVD threshold			
		10^{-14}	10^{-6}	10^{-4}	10^{-1}
BARTHT1A	(its)conv	(94)	(97)	0.1E+01	0.1E+01
	$\|\|(LU)^{-1}e\|\|_\infty$	0.11E+08	0.11E+08	0.51E+07	0.14E+05
BARTHT2A	(its)conv	(545)	(449)	0.1E+01	0.1E+01
	$\|\|(LU)^{-1}e\|\|_\infty$	0.11E+08	0.11E+08	0.51E+07	0.14E+05
WIGTO966	(its)conv	0.7E+00	0.7E+00	0.7E+00	(50)
	$\|\|(LU)^{-1}e\|\|_\infty$	0.15E+09	0.15E+09	0.14E+08	0.72E+05
BBMAT	(its)conv	0.7E+02	0.7E+02	0.1E+01	0.1E+01
	$\|\|(LU)^{-1}e\|\|_\infty$	0.24E+14	0.17E+12	0.28E+05	0.19E+03
	ILUnnzb	209 006	209 006	209 006	209 006

Table 18: Diagonal perturbation with BILU(0)

(or no) information from blocks far from the diagonal. The matrix can also be made more diagonally dominant by increasing the absolute value of the diagonal entries. This changes the matrix, and it is done for the purpose of calculating the preconditioner only.

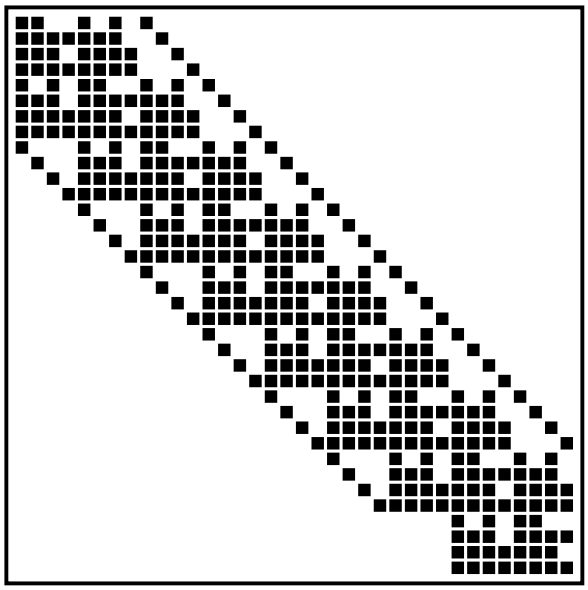
The decision on which modifications to use for BBMAT is guided by observing its non-zero pattern. This is shown in figure 3 for the complete matrix, and for 250x250, and 36x36 submatrices. From the figure it can be seen that BBMAT has a small block size of 4, and a large block size of 232. There are two bands above the diagonal band, and two bands below. The number of entries separating bands is 232. The entries in the top right-hand corner, and the bottom left-hand corner of the complete matrix are from some kind of periodic boundary condition.

Experiments on BBMAT with banded ILU show that for band size small enough to exclude contributions from the off diagonal bands (this is band size ≤ 228), the condition number estimate $\|(LU)^{-1}e\|_\infty$ has values less than 10^4, and convergence stalls at about 600 GMRES steps, with a reduction in residual norm of 0.016. Convergence was similar for a range of bandwidths from 16 to 288. For a band width of 232, which includes entries from the first off-diagonal band, the condition number estimate $\|(LU)^{-1}e\|_\infty$ is 0.2×10^{24}. Thus including off-diagonal bands in calculating the ILU preconditioner results in unacceptably high condition number.

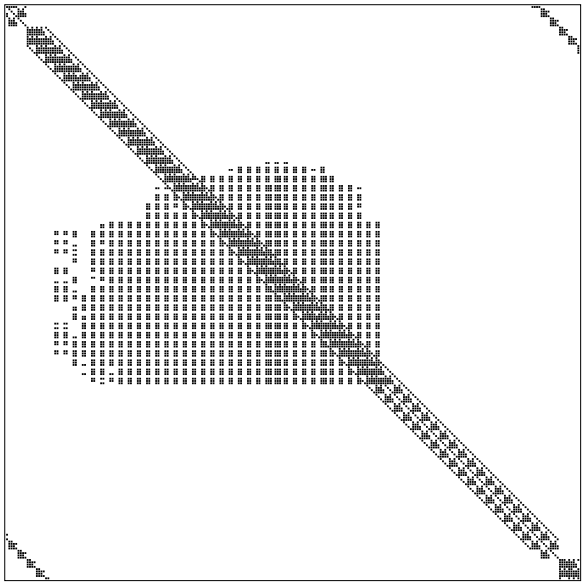
Block diagonal and BSSOR were tried using dense matrix LU to invert the diagonal blocks of size 4, 8, 12, and 16. The BSSOR relaxation parameter was set to 0.5, and a single BSSOR step was used. Convergence stalled at about 200 GMRES iterations. The best convergence was a reduction in residual norm of 0.015 for BSSOR with block size 16. The a BSSOR relaxation parameter was set to 0.5, and a single BSSOR step was used. The condition number $\|(LU)^{-1}e\|_\infty$ for BSSOR preconditioning was less than 10^4.

Block diagonal and block SSOR were tried with large block size (232 or 464), using ILUTP to invert the diagonal blocks. Here convergence stalled at about 600 GMRES steps, with a reduction in residual norm of 0.016. The tests using large block size and ILUTP are slower that those for small block size and dense LU.

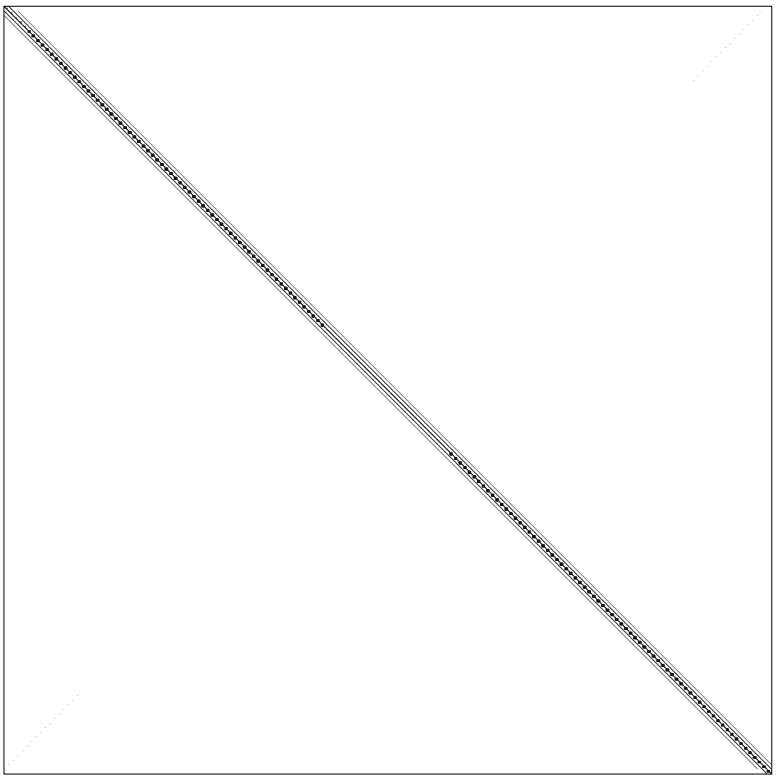
Tests with increasing the diagonal dominance for BSSOR resulted in a reduced condition



36×36 submatrix



250×250 submatrix



complete matrix

Figure 3: Non-zero pattern for BBMAT

number estimator $\|(LU)^{-1}e\|_\infty$, but worse convergence.

For problems where GMRES stalls, as in the tests above, a remedy which sometimes breaks the stalling behavior is deflation. Here eigenvectors of the preconditioned matrix are estimated as GMRES proceeds, and they are added to the Krylov subspace. Deflation was tried with the preconditioners above, adding 4 estimated eigenvectors to the Krylov subspace (which has a total size of 50). The best results were obtained for BSSOR preconditioning, using dense matrix techniques to invert the diagonal blocks. Convergence results are given in Table 19, and convergence history for BSSOR with block size 16 is shown in figure 4. The improved performance for the larger block size does not justify the increased memory required, so for BBMAT the best preconditioner with deflation is BSSOR with block size 16.

block size	$\|(LU)^{-1}e\|_\infty$	time C90 sec	number iterations	reduction in residual
16	0.50×10^2	702	2400	0.19×10^{-6}
32	0.11×10^3	744	2400	0.48×10^{-3}
48	0.14×10^3	712	2400	0.24×10^{-7}
64	0.16×10^3	837	2400	0.17×10^{-7}
116	0.30×10^3	696	2400	0.15×10^{-7}
232	0.42×10^3	1197	2023	0.91×10^{-8}

Table 19: Dense BSSOR with deflation for BBMAT

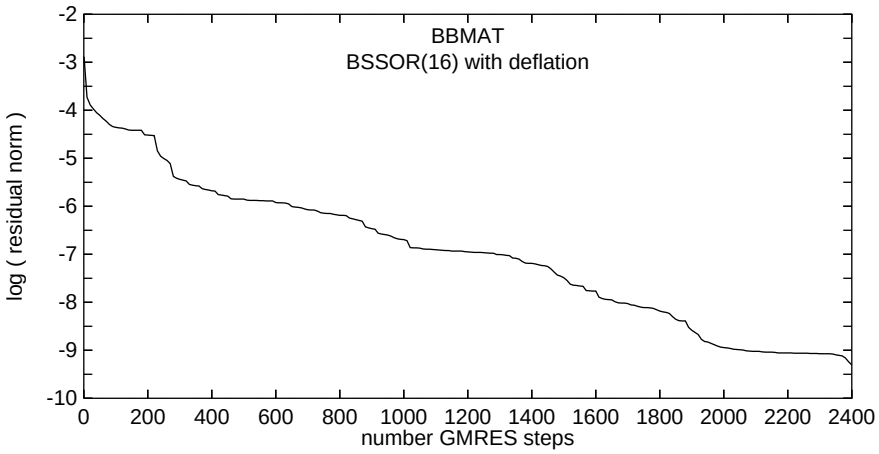


Figure 4: Convergence history for solving BBMAT using BSSOR and deflation

6 Conclusion

In the numerical tests section of this report, point and block preconditioners with level of fill, threshold, and diagonal compensation dropping strategies were tried. No one preconditioner

was the best for all matrices. Rougly speaking, it was found that preconditioners gave similar performance for a similar amount of fill-in.

In an ILU factorization scheme the factors L and U are generated such that $A = LU - E$, where E represents the matrix of fill-ins that are discarded during the factorization process. The aim is to make LU as accurate a representation of A as possible, and this is to be achieved with as few non zero entries as possible. It is also important that $(LU)^{-1}$ have a reasonable condition number. Often, when A is ill-conditioned and indefinite, the matrix LU may be much worse conditioned than A itself. As the factorization becomes more accurate, the likelihood of obtaining an unstable factorization seems to increase. A useful estimate of the condition number of $(LU)^{-1}$ is $\|(LU)^{-1}e\|_\infty$, where e is a vector of all ones.

An approach which was used for matrices which have highly ill-conditioned factors was to give special treatment to entries close to the diagonal, to enhance their contribution to the ILU factors. The matrices from CFD applications often have a banded non-zero structure, with a diagonal band, and typically one or two bands above the diagonal, and one or two bands below. A way to decrease the condition number for such matrices is to only consider entries in the diagonal band when calculating the ILU preconditioner, or to only consider entries within diagonal blocks, small enough to exclude contributions from off diagonal bands, this is equivalent to block diagonal preconditioning. These approaches were found to work best in one case (matrix BBMAT). Another approach is to increase diagonal dominance by increasing the absolute value of the diagonal elements, or by perturbing the diagonal blocks. Perturbation of diagonal blocks was found to be effective for the matrix WIGTO966.

As is known, point and block factorizations based on level-of-fill are identical if zero entries in blocks are filled in. This is confirmed by the experiments. The point and block preconditioners based on threshold calculate fill-in pattern as the factorization progresses, by dropping entries or blocks with small magnitude, and limiting the number of entries or blocks per row in the ILU factors. It was found that the block version of threshold preconditioning has better convergence that the point version in general. This may be due to the coupling between entries within a block, making it better to drop or keep complete blocks, rather than individual entries.

The block preconditioners for both level of fill and threshold have a leaner data structure and require less storage for pointer arrays than the point versions. Calculating the block structure requires less computational effort then calculating the point structure, because there are less blocks in the matrix than entries. The block preconditioners are usually faster, but this depends on the speedup obtained from performing block row operations rather than row operations, versus the extra time taken inverting individual blocks.

Acknowledgement The Minnesota Supercomputer Institute has provided computer facilities and an excellent research environment to conduct this research. We wish to thank Tim Barth from NASA Ames for supplying us with some of the matrices used in this work.

References

[1] Andrew Chapman, and Yousef Saad. Deflated and augmented Krylov subspace techniques *Technical Report UMSI 95/181*, Minnesota Supercomputer Institute, 1995.

[2] Edmond Chow, and Mike Heroux. An object oriented framework for block preconditioning. *Technical Report UMSI 95/216*, Minnesota Supercomputer Institute, 1995.

[3] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices. Technical Report UMSI 94-101, University of Minnesota Supercomputer Institute, Minneapolis,MN 55415, May 1994.

[4] E. Chow and Y. Saad. Stabilized block ILU preconditioners. Technical Report, University of Minnesota Supercomputer Institute, Minneapolis, MN 55415, In preparation.

[5] J. A. Meijerink, and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31(137):148-162,1977.

[6] R. B. Morgan A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Analysis and Applications*, 16, 1154-1171, 1996.

[7] W. H. Press, B. P. Flannnery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, 1986.

[8] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Halstead Press, New York, 1992.

[9] Y. Saad A dual threshold incomplete LU factorization. *Num. Lin. Alg. Appl.*, 1 (1994), pp. 387-402.

[10] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS publishing, New York, 1996.

[11] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7, 856–869, 1986.

[12] Yousef Saad and Kesheng Wu. DQGMRES: a quasi-minimal residual algorithm based on incomplete orthogonalization *Technical Report UMSI 93/131*, Minnesota Supercomputer Institute, 1993.

[13] J. W. Watts-III. A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation. *Society of Petroleum Engineer Journal*, 21:345-353, 1981.