ACCELERATED JACOBI ITERATIONS FOR BIDIAGONAL AND SPARSE TRIANGULAR MATRICES

THOMAS K. HUCKLE *

May 7, 2019

Abstract. In many applications a sparse linear system of equations $Ax = b$ has to be solved. For applying iterative solvers like preconditioned conjugate gradient (pcg) or GMRES, effective preconditioners are necessary, e.g. Jacobi, Gauss-Seidel, or incomplete LU factorization (ILU). Often, effective preconditioners are given via sparse triangular matrices L, that have to be solved in every iteration step. Recent work by Edmond Chow introduced an easy to parallelize fixed-point iteration for computing approximations to (I)LU factorizations. Therefore, the aching handicap in parallel solution methods for sparse matrices is the solving of sparse triangular systems, e.g. bidiagonal matrices. In a parallel environment direct solvers can take only restricted advantage of parallelism. Therefore, in this paper we develop a fast iterative solution method for sparse triangular matrices. In contrast to direct solvers for triangular matrices L like graph-based methods, sparse factorization methods, or Sherman-Morrison-Woodbury, here we want to consider stationary Jacobi iterations. In its original form the Jacobi iteration for ill-conditioned matrices can lead to very slow convergence. Therefore, we introduce different acceleration tools like preconditioning (block Jacobi and Incomplete Sparse Approximate Inverse ISAI), and a recursive acceleration of the Jacobi method. Here the Neumann series is replaced by the Euler expansion (see [4, 19, 8]). This is derived by a recursive computation of the Neumann series using powers of the initial Jacobi iteration matrix. The goal is to shift the major part of the operations from cheap but numerous iteration steps to better parallelizable cheap and sparse matrix-matrix products reducing the number of necessary iterations considerably, e.g. to less than $log_2(n)$ for an $n \times n$ matrix.

Key words. Jacobi method, preconditioning, sparse triangular linear system, bidiagonal linear system, parallel computing

AMS subject classifications. 65F08, 65Y05, 65F50, 65F10

1. Introduction. For solving sparse linear systems of equations $Ax = b$ for matrix A there exist direct solution methods related to the Gaussian Elimination like the Cholesky or the LU factorization (see [1, 2, 13, 14, 15, 18, 20, 22, 24]) or iterative solution methods like the conjugate gradient method (cg) or GMRES [21]. Here, the iterative methods usually need a preconditioner P or M in the form $P^{-1}Ax = P^{-1}b$ or $MAx = Mb$ (for left preconditioning) in order to derive satisfying convergence. Here, P is seen as an approximation on A directly while M is an approximation on A^{-1}. Especially on a highly parallel computing architecture the preconditioner should additionally be easy to derive and to apply. Preconditioners like Gauss-Seidel (GS) or Incomplete LU decomposition (ILU) are efficient to set up in parallel, e.g. via the iterative fixed-point iteration by Chow [9]. But often the remaining bottleneck in direct or iterative solvers is the efficient parallel solution of sparse triangular systems resulting from factorized representations or approximations of A. Note, that the derived iterative solvers can also be applied for (twisted) bidiagonal systems in connection with eigenvector solvers [23].

Therefore, in this paper we concentrate on efficient parallel solution methods for sparse triangular and esp. bidiagonal matrices $Lx = b$. Because Krylov methods are not very efficient for triangular systems, we consider the stationary Jacobi method. To derive faster convergence and allow better parallel efficiency we include

* Department of Computer Science, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Munich, Germany (huckle@in.tum.de).

n	Jacobi	BiCGSTAB	GMRES	GMRES(50)	GMRES(20)
100	99	98	99	302	253
200	199	*	197	599	387
400	399	*	394	695	565

TABLE 2.1

Iteration count for different iterative solvers for ill-conditioned $L = tridiag(-1, 1, 0)$, random rhs, relative error 10^{-6}.

- a preconditioner like block Jacobi or incomplete sparse approximate inverses ISAI to L [3],
- a recursive acceleration of convergence by representing the Neumann series via the multiplicative Euler expansion - computing every second, fourth, eighth,.. , 2^k-th partial sum of the Neumann series with $L_0 = E - L$, E the identity matrix:

$$E + L_0 + L_0^2 + L_0^3 + L_0^4 ... = ...(E + L_0^4)(E + L_0^2)(E + L_0) \ .$$

This has the advantage of replacing vector updates by parallelizable sparse matrix-matrix products, giving more and more exact entries of the solution vector in every step, and delivering the exact solution in the worst case after $\log_2(n)$ iterations.

In the case of L a bidiagonal matrix, it turns out that this acceleration of the Jacobi iteration comes without additional costs compared to the original Jacobi method, and can be applied until convergence because the necessary matrix-matrix products are nothing else than simple vector operations. In case of a general sparse triangular matrix the iteration matrix is getting more dense in every step, thus prohibiting the computation of the matrix products and forcing the algorithm to stop after some recursive steps and to proceed with the Jbasic acobi iteration based on the last computed iteration matrix.

In section 2 we describe the modification of the Jacobi method for sparse triangular matrices. In section 3 we analyze the special case of bidiagonal matrices. In section 4 we present numerical results. Section 5 introduces a new hybrid preconditioner, and section 6 contains the conclusions.

2. Jacobi method for triangular matrices. We consider a linear system of equations of the form $Lx = b$ with L being sparse triangular or even bidiagonal. To simplify the equations, without loss of generality in the following we assume $diag(L) = E$ the identity matrix. In view of the triangular structure of L, reasonable choice of iterative solvers are the stationary Jacobi method, or Krylov methods like GMRES, BiCGSTAB, or QMR [21, 5]. Numerical tests reveal (see tables 2.1 and 2.2), that the Krylov methods take the same number of iterations, resp. matrix-vector multiplications, or even more, as the basic Jacobi iteration:

$$x^{(k+1)} = b + (E - L)x^{(k)} = b + L_0 x^{(k)} = x^{(k)} + (b - Lx^{(k)}) \ .$$

Furthermore, the Jacobi iteration leads to an iteration matrix L_0 with $diag(L_0) = 0$. Hence, L_0 has spectral radius zero and the Jacobi iteration is guaranteed to converge - at least in exact arithmetic. In the contrary, GMRES would use dense Hessenberg matrices of growing size, and BiCGSTAB needs even two matrix-vector multiplications per step. Unfortunately, the Jacobi method for ill-conditioned matrices A can be painfully slow. Furthermore, for ill-conditioned L the convergence is guaranteed only theoretically, in practical runs in view of the large condition of L_0 the norm of the

n	Jacobi	BiCGSTAB	GMRES	GMRES(50)	GMRES(20)
100	99	69	96	115	110
200	127	67	121	122	123
400	129	70,5	122	122	122

TABLE 2.2

Iteration count for different iterative solvers for well-conditioned $L = tridiag(-0.9, 1, 0)$, random rhs, relative error 10^{-6}.

residuals will be growing. E.g., for $L = tridiag(1.1, 1, 0)$ and $n = 100$, $b = ones(n)/\sqrt{n}$ the Jacobi method leads to growing residuals, only after 99 iterations the last iteration results in a small residual. Therefore it is necessary to accelerate the iteration by preconditioning. Also here, only a few choices are reasonable. Obviously, Gauss-Seidel and ILU are not meaningful. There remains the block Jacobi preconditioner or sparse approximate inverse type methods. In the block Jacobi method we consider not only the diagonal entries for preconditioning (note that $diag(L) = E$ would be the identity and rather useless as preconditioner), but a block diagonal structure B with blocks of prescribed size. So for general A we set $P = blockdiag(A)$, $M = P^{-1}$ which leads to the formula $(AM = E)_B$ on the block diagonal pattern. (Note, that the main diagonal blocks of the preconditioner are the inverses of the main diagonal blocks of L.) This equation proves that the iteration matrix for the block Jacobi preconditioned system in the case of a triangular matrix $(LM - E)_B$ is zero on the block pattern B which again guarantees convergence.

In the sparse approximate inverse approach (SAI) (see [6, 7, 10, 11, 12]) the sparse preconditioner is derived by solving in the Frobenius norm $\min_S ||AM - E||_F^2$ for given sparse matrix A, and sparsity pattern S for M. This minimization can be done columnwise in parallel

$$\min \|AM - E\|_F^2 \;=\; \sum_{k=1}^{n} \min \|AM_k - e_k\|^2 \;.$$

Let us denote the allowed nonzero entries in M_k with J_k. In view of the sparsity of M_k this minimization then reduces to $\min ||A(:, J_k)M_k(J_k) - e_k||_2^2$. Taking into account also the sparsity of A, most of the rows in $A(:, J_k)$ are zero. Denoting the nonzero rows with I_k, the minimization reduces to the small Least Squares problem

$$\min ||A(I_k, J_k)M_k(J_k) - e_k(I_k)||_2^2 \;.$$

This approach is embarrassingly parallel but the derived SAI preconditioner often does not reduce the number of iteration steps in the same way as direct preconditioners like ILU.

In the special case of triangular L, SAI leads to the triangular Least Squares problem

$$min||L(I_k, J_k)M_k(J_k) - e_k(I_k)||_2^2 \;.$$

In this special case it is also possible to replace the Least Squares problem in rectangular $L(I_k, J_k)$ by the smaller square problem

$$\min ||L(J_k, J_k)M_k(J_k) - e_k(J_k)||_2^2 \;,$$

resp. the small system of linear equations

$$L(J_k, J_k)M_k(J_k) = e_k(J_k) \;.$$

n	None	ISAI k=1	k=10	SPAI k=1	k=10	Block-Jac m=2	m=10
100	99	49	9	100	31	49	9
200	199	99	18	200	54	99	18
400	399	199	36	400	97	199	36

TABLE 2.3

Iteration count for preconditioned Jacobi solvers with different preconditioner for $L1 = tridiag(-1,1)$*, random rhs, relative error* 10^{-6}*. The used preconditioner are ISAI and SAI to pattern* $abs(L)^k$ *and Block Jacobi with block size* m*.*

n	None	ISAI k=1	k=10	SPAI k=1	k=10	Block-Jac m=2	m=10
10*10	18	9	1	25	6	13	9
20*20	38	19	3	43	10	28	27
40*40	78	39	7	78	18	58	49

TABLE 2.4

Iteration count for preconditioned Jacobi solvers with different preconditioner for $L = (kron(L1,E) + kron(E,L1))/2$*, random rhs, relative error* 10^{-6}*. The used preconditioner are ISAI abd SAI to pattern* $abs(L)^k$ *and Block Jacobi with block size* m*.*

For triangular, nonsingular L, the square submatrix is also triangular and nonsingular. In the following we call this preconditioner ISAI for Incomplete Sparse Approximate Inverse [3, 6]. Like in the block Jacobi case this preconditioner fulfills $(LM - E)_S = 0$ on pattern S which again guarantees convergence of the preconditioned Jacobi iteration as long as the pattern S contains the main diagonal. This property does not hold for the original SAI preconditioner. So ISAI can be seen as a simplification of the Sparse Approximate Inverse replacing the Least Squares condition by an incompleteness condition $(AM = E)_S$ similar to the ILU incompleteness condition $(A - LU)_S = 0$ on pattern S. The block Jacobi preconditioner is a special case of ISAI for S chosen as block diagonal pattern. Including such a preconditioner reduces the number of iterations introducing operations that are fully parallel like independent solution of many small linear systems and matrix-vector products. The preconditioned Jacobi iteration is described by

$$x^{(k+1)} = Mb + (I - ML)x^{(k)} = x^{(k)} + M(b - Lx^{(k)})$$

for left preconditioning, resp. for right preconditioning

$$y^{(k+1)} = b + (I - LM)y^{(k)} = y^{(k)} + (b - LMy^{(k)})\,, x^{(k)} = My^{(k)}\,.$$

For right preconditioning ISAI leads to $(LM - I)_S = 0$ on pattern S, resp. in left preconditioning $(ML - I)_S = 0$ on pattern S.

THEOREM 2.1. *For triangular* L *the ISAI peconditioner leads to an iteration matrix* $L_0 = E - LM$ *for left preconditioning, resp.* $L_0 = E - ML$ *for right preconditioning. Therefore the preconditioned Jacobi iteration is based on a triangular iteration matrix* L_0 *with zero eigenvalues that is contracting in an appropriate norm, and convergence is guaranteed - at least in exact arithmetic.*

The comparison of table 2.3 and 2.4 shows that ISAI leads to the fewest number of iterations.

In view of the slow convergence for ill-conditioned matrices, we introduce an additional acceleration method. The basic Jacobi iteration with initial guess $b_0 = b$ generates approximations given by the Neumann series

$$x^{(k)} = b + L_0 b + L_0^2 b + ... + L_0^k b$$

with

$$L^{-1}b = (E - L_0)^{-1}b = \sum_{j=0}^{\infty} L_0^j b \ .$$

This Neumann series can also be expressed by the Euler expansion

$$\sum_{j=0}^{\infty} L_0^j b = ...(E + L_0^8)(E + L_0^4)(E + L_0^2)(E + L_0)b \ .$$

For the special bidiagonal case

$$L = \begin{pmatrix} 1 & & & & \\ d_2 & 1 & & & \\ & d_3 & 1 & & \\ & & . & . & \\ & & & d_n & 1 \end{pmatrix} ,$$

$L_0 = E - L$ has nonzeros only at the lower subdiagonal entries $-d_2, ..., -d_n$. Therefore, the powers L_0^k are easy to compute and each power can be fully described with exactly one vector of nonzero entries at the k-th subdiagonal (starting with $l_{k+1,1}$) and zeros elsewhere (denoting the main diagonal with $k = 0$). Especially easy to compute are the powers $L_{0,k} := L_0^{2^k}$ that we need in the Euler expansion. To this aim, we define $L_{0,0} = L_0$ and for $k = 1, 2, ...$: $L_{0,k} = L_{0,k-1} * L_{0,k-1}$. Then, $L_{0,2^k}$ has only nonzero entries on and below the 2^k-te subdiagonal, and gets zero for $k > \log_2(n)$.

For general sparse triangular L, the matrices $L_{0,k}$ have a similar pattern but allowing also nonzero entries below the k-th subdiagonal. In general, it holds $L_{0,\lceil \log_2(n) \rceil} = 0$.

To take advantage of the above used Euler expansion we recursively merge in the Jacobi iterations two steps into one allowing powers of L_0:

$$\begin{aligned}
x^{(0)} &= b = b_0 \\
x^{(1)} &= b + L_0 x^{(0)} = b_0 + L_0 x^{(0)} \\
x^{(2)} &= b + L_0 x^{(1)} = b + L_0(b + L_0 x^{(0)}) = b + L_0 b + L_0^2 b = b_1 + L_0^2 x^{(0)} \\
x^{(4)} &= b + L_0 b + L_0^2 b + L_0^3 b + L_0^4 x^{(0)} = b_2 + L_0^4 x^{(0)} \\
x^{(8)} &= b_3 + L_0^8 x^{(0)} \\
&\vdots \\
x^{(2^k)} &= b_k + L_0^{2^k} x^{(0)}
\end{aligned}$$

Here, $b_k = \sum_{j=0}^{k} L_0^k b$ with $b_{k+1} = (E + L_0^{2^k})b_k$. This leads to two different accelerations methods for the Jacobi method. So Algorithm 1 computes the Neumann series via Euler expansion starting with $x^{(0)} = b$ and left preconditioner M until the residual is less than a given accuracy *ep*. In many cases the matrices L_0 might get too dense. Therefore, in this case one can stop the recursion and proceed with the original Jacobi iteration based on the last computed iteration matrix L_0 (the 2^k-th power of the original iteration matrix), until convergence. This is described by Algorithm 2. Here, *itmax* denotes the maximum number of overall iterations:

Algorithm 1 Accelerated left preconditioned Jacobi method via Neumann series.

$b_0 = Mb$; $c = b_0$;
$it = 0$; $er = 1$;
$L_0 = E - M \cdot L$;
while $er > ep$ and $it < log2(n)$ **do**
  $b_0 = b_0 + L_0 b_0$;
  $L_0 = L_0 \cdot L_0$;
  $er = ||Lb_0 - c||$; $it = it + 1$;
end while

Algorithm 2 Accelerated left preconditioned Jacobi method with restricted recursion.

$b_0 = Mb$; $c = b_0$;
$it = 0$; $er = 1$;
$L_0 = E - M \cdot L$;
while $er > ep$ and $it < itmax$ **do**
  if L_0 is still sparse enough **then**
    $b_0 = b_0 + L_0 b_0$;
    $x = b_0$;
    $L_0 = L_0 \cdot L_0$;
    if $er = ||Lx - c|| < ep$ **then**
      $break$;
    end if
    $it = it + 1$;
  else
    $x = b_0 + L_0 x$;
    $er = ||Lx - c||$;
    $it = it + 1$;
  end if
end while

Combining the accelerated Jacobi algorithms with the ISAI preconditioner leads to an iteration matrix with zeros on the pattern S. In the special case that we include in the pattern a certain bandwidth s, the central diagonals will be zero and nonzeros appear only on and below the s-th subdiagonal. Therefore, algorithms 1 and 2 already start with a matrix of this form. This proves

THEOREM 2.2. *If the accelerated Jacobi iteration is applied on an ISAI preconditioned triangular matrix with a pattern S that includes a bandwidth s, then for $k = \lceil \log_2(n/(s-1)) \rceil$ the k-th power of the iteration matrix will be zero and therefore the method converges at least in k steps.*

Furthermore, for right preconditioned L with pattern S, the nonzero entries in $E - LM$ are columnwise below the nonzero entries of L. Therefore, the preconditioner moves the nonzero pattern down-leftwards.

The accelerated Jacobi method - similar to the Jacobi iteration - has the nice property that after the first step the first entry of the approximate solution vector is exact, after the second step the two upper entries are exact, and after k steps the first 2^{k-1} entries are exact.

The ISAI preconditioner in connection with sparse triangular L is easy to compute. This is shown by the following

Theorem 2.3. *Let S_L be the pattern of the given triangular matrix L with all diagonal entries 1 and $L = E - L_0$. The triangular ISAI right preconditioner M is defined with pattern S where $S_L \subseteq S$ of the form $M = E + M_0$. Then for all entries $(i, j), i \neq j$, in the pattern $S \backslash S(L_0 M_0)$ it holds $M_{i,j} = -L_{i,j}$.*

For the proof we only have to consider $LM = (E - L_0)(E + M_0) = E - L_0 + M_0 - L_0 M_0 = E$ on pattern S. Furthermore, for the pattern of $L_0 M_0$ it holds:

Theorem 2.4. *The nonzero entries in the pattern of LM different from the diagonal entries are always columnwise lower as the entries of M_0 and rowwise left to the entries of L_0.*

This theorem is a direct consequence of the directed graph for L_0 and M_0 and the fact that in the graph fill-in caused by the product refers to paths of length 2 in the combined graphs. So by preconditioning the nonzero pattern - besides the main diagonal - is shifted to the left down corner. The generated fill-in depends on the degree of the nodes in the two graphs.

Because the ISAI preconditioners ensures that $LM = E$ on pattern S, the preconditioned matrix has nonzero entries on the main diagonal, and otherwise only inside the pattern of $S(L_0 M_0) \backslash S$.

Considering the outer bandwidth of strictly triangular matrices it is easy to proof the following

Theorem 2.5. *Let L_0 and M_0 be strictly lower triangular with nonzero entries on the k-th, resp. the m-th subdiagonal and below. Then $L_0 M_0$ has nonzero entries only below the $k + m - 1$-th diagonal.*

$$L_0 = \begin{pmatrix} 0 &&&&&&& \\ * & 0 &&&&&& \\ & * & 0 &&&&& \\ && * & 0 &&&& \\ &&& * & 0 &&& \\ &&&& * & 0 && \\ &&&&& * & 0 & \\ &&&&&& * & 0 \end{pmatrix}, \quad L_0^2 = \begin{pmatrix} 0 &&&&&&& \\ 0 & 0 &&&&&& \\ * & 0 & 0 &&&&& \\ & * & 0 & 0 &&&& \\ && * & 0 & 0 &&& \\ &&& * & 0 & 0 && \\ &&&& * & 0 & 0 & \\ &&&&& * & 0 & 0 \end{pmatrix}, \quad L_0^4 = \begin{pmatrix} 0 &&&&&&& \\ 0 & 0 &&&&&& \\ 0 & 0 & 0 &&&&& \\ 0 & 0 & 0 & 0 &&&& \\ * & 0 & 0 & 0 & 0 &&& \\ & * & 0 & 0 & 0 & 0 && \\ && * & 0 & 0 & 0 & 0 & \\ &&& * & 0 & 0 & 0 & 0 \end{pmatrix}, \quad L_0^8 = \begin{pmatrix} 0 &&&&&&& \\ 0 & 0 &&&&&& \\ 0 & 0 & 0 &&&&& \\ 0 & 0 & 0 & 0 &&&& \\ 0 & 0 & 0 & 0 & 0 &&& \\ 0 & 0 & 0 & 0 & 0 & 0 && \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 2.1. *Pattern of powers of iteration matrix $L_0^{2^k}$.*

The change in the pattern of the iteration matrix is shown in fig. 2.1.

For banded matrices the ISAI preconditioner does not lead to increasing number of nonzero entries in the preconditioned matrix LM. It holds

Theorem 2.6. *Let L be triangular with band width k, and M the resulting ISAI preconditioner with band width $m \geq k$. Then $LM - E$ has only nonzero entries from subdiagonal $m + 1$ to $m + k$.*

As a sign of warning, table 2.5 shows the limits of iterative solvers for ill-conditioned triangular matrices. A loss of convergence can be observed for both preconditioned Jacobi and recursive Jacobi with growing n. To avoid this behavior the application of iterative solvers should be restricted to matrices L satisfying a decay property in the entries of L such that $||L_0||$ is small (< 1). This excludes a wide class of examples. Numerical tests reveal that also application of GMRES or BiCGSTAB does not help in these cases. Also direct solvers will result in large relative errors in these cases. The tables 2.6 and 2.7 show the lack of accuracy, resp. the stagnation, for iterative and direct solvers for ill-conditioned bidiagonal matrices.

n	Jac. k=1	k=2	k=4	k=8	16
20	9	6	3	2	1
40	19	13	7	4	2
80	39	26	15	8	4
160	79	53	31	17	9
320	*	*	*	*	*
n	rec. Jac. k=1	k=2	k=4	k=8	k=16
20	4	3	2	2	1
40	5	4	3	3	2
80	6	5	4	4	3
160	7	6	5	5	4
320	*	*	*	*	*

TABLE 2.5

Iteration count for different preconditioned iterative solvers for ill-conditioned $L = tridiag(1.1, 1, 0)$, rhs $ones(n)/\sqrt{n}$, relative error 10^{-8}. As preconditioner, ISAI is applied with bandwidth k.

k	0	1	2	3
BiCGSTAB	0.0167 (2)	0.322 (1)	0.0376 (8)	0.267 (1)
GMRES(50)	0.0109 (6,50)	0.326 (4,50)	0.0326 (7,50)	0.3047 (4,50)
rec. Jac.	0.679 (9)	2.478 (9)	0.879 (9)	1.622 (9)
Jacobi	0.1218 (400)	0.278 (400)	0.1118 (400)	0.249 (400)
Gauss	0.1218	0.1218	0.1218	0.1218

TABLE 2.6

Iteration count for different preconditioned iterative solvers for ill-conditioned $L = tridiag(1.1, 1, 0)$, rhs $ones(n)/\sqrt{n}$, relative error 10^{-6}, and $n = 400$. As preconditioner, ISAI is applied with bandwidth k. Comparison of different solvers: BiCGSTAB, GMRES(50), recursive Jacobi, Jacobi, and Gauss elimination as direct solver. The entries in the table are the norm of the last residual and the number of iterations.

3. Accelerated Jacobi iteration for preconditioned bidiagonal matrices. Algorithm 1 can be formulated very efficiently for bidiagonal L. Here, the matrix powers are nonzero on one subdiagonal only and Algorithm 1 can be rewritten in the form of Algorithm 3. In the special case that the subdiagonal entries are all -1 we do not have to store these subdiagonal entries and Algorithm 1, resp. 3, simplify to Algorithm 4.

Algorithm 3 Accelerated unpreconditioned Jacobi method for bidiagonal L.

$b_0 = b$;
$d = (-d_2, ..., -d_n)$;
for $k = 1 : log2(n)$ **do**
  $b_0(2^k + 1 : n) = b_0(2^k + 1 : n) + d(1 : n - 2^k). * b_0(1 : n - 2^k)$;
  $d(1 : n - 2^k) = d(1 : n - 2^k). * d(2^k + 1 : n)$;
  if $||Lb_0 - b|| < ep$ **then**
    break;
  end if
end for

k	0	1	2	3
BiCGSTAB	0.015 (4)	0.35 (1)	0.037 (3)	0.288 (1)
GMRES(50)	0.010 (8,50)	0.346 (6,50)	0.031 (3,50)	0.32 (8,50)
rec. Jac.	0.031 (9)	0.075 (9)	0.00786 (9)	0.03777 (9)
Jacobi	0.0027 (400)	0.0376 (400)	0.0026 (400)	0.0129 (400)
Gauss	0.0071	0.0071	0.0071	0.0071

TABLE 2.7

Iteration count for different preconditioned iterative solvers for ill-conditioned $L = tridiag(1.09, 1, 0)$, *rhs* $ones(n)/\sqrt{n}$, *relative error* 10^{-6}, *and* $n = 400$. *As preconditioner, ISAI is applied with bandwidth k. Comparison of different solvers: BiCGSTAB, GMRES(50), recursive Jacobi, Jacobi, and Gauss elimination as direct solver. The entries in the table are the norm of the last residual and the number of iterations.*

Algorithm 4 Accelerated unpreconditioned Jacobi method for bidiagonal $L = tridiag(-1, 1, 0)$.

```
b_0 = b;
for k = 1 : log2(n) do
    b_0(2^k + 1 : n) = b_0(2^k + 1 : n) + b_0(1 : n - 2^k);
    if ||L b_0 - b|| < ep then
        break;
    end if
end for
```

To include also preconditioning, let us consider

$$L = \begin{pmatrix} 1 & & & & \\ d_2 & 1 & & & \\ & d_3 & 1 & & \\ & & \cdot & \cdot & \\ & & & d_n & 1 \end{pmatrix}, \; L_0 = \begin{pmatrix} 0 & & & & \\ -d_2 & 0 & & & \\ & -d_3 & 0 & & \\ & & \cdot & \cdot & \\ & & & -d_n & 0 \end{pmatrix},$$

and the ISAI preconditioner is then derived by the entries of the inverse L^{-1} that coincide with the chosen pattern S. The inverse is given by

$$L^{-1} = \begin{pmatrix} 1 & & & & & & \\ -d_2 & 1 & & & & & \\ d_2 d_3 & -d_3 & 1 & & & & \\ \vdots & \ddots & \ddots & \ddots & & & \\ -(-1)^k d_2 \cdots d_k & \cdots & d_{k-1} d_k & -d_k & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ -(-1)^n d_2 \cdots d_n & \cdots & \cdots & \cdots & d_{n-1} d_n & -d_n & 1 \end{pmatrix}.$$

Hence, the ISAI preconditioner is given by $(L^{-1})_S$, in the banded case by

$$M = \begin{pmatrix} 1 & & & & & & \\ -d_2 & 1 & & & & & \\ d_2 d_3 & -d_3 & 1 & & & & \\ \vdots & \ddots & \ddots & \ddots & & & \\ -(-1)^k d_2 \cdots d_k & \cdots & d_{k-1} d_k & -d_k & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -(-1)^k d_{n-k+2} \cdots d_n & \cdots & d_{n-1} d_n & -d_n & 1 \end{pmatrix}.$$

k/n	100	200	400	800	1600	3200
Jac. $k=0$	99	199	399	799	*	*
rec.Jac. $k=0$	*	*	*	*	*	*
Jac. $k=1$	33	66	133	266	533	>600
rec.Jac. $k=1$	6	7	8	9	10	11
Jac. $k=2$	16	33	66	133	266	>500
rec.Jac. $k=2$	5	6	7	8	9	10
Jac. $k=3$	14	28	*	*	*	*
rec.Jac. $k=3$	4	5	*	*	*	*

TABLE 4.1

Iteration count for preconditioned Jacobi solvers with different preconditioner for $L = pentdiag(1,1,1,0,0)$*, rhs* $ones(n)/\sqrt{n}$*, relative error* 10^{-6}*. The used preconditioner are ISAI to pattern* $abs(L)^k$*.*

k/n	100	200	400	800	1600
Jac. $k=0$	99	199	399	799	>1000
rec.Jac. $k=0$	*	*	*	*	*
Jac. $k=1$	24	45	99	399	533
rec.Jac. $k=1$	5	6	7	8	9
Jac. $k=2$	12	24	49	99	199
rec.Jac. $k=2$	4	5	6	7	8
Jac. $k=3$	8	16	33	66	133
rec.Jac. $k=3$	4	5	6	7	8

TABLE 4.2

Iteration count for preconditioned Jacobi solvers with different preconditioner for $L = septadiag(1,1,1,1,0,0)$*, rhs* $ones(n)/\sqrt{n}$*, relative error* 10^{-6}*. The used preconditioner are ISAI to pattern* $abs(L)^k$*.*

Then, we have to compute Mb and the resulting iteration matrix for preconditioned Jacobi method is given by

$$E - ML = \begin{pmatrix} 0 & & & & & 0 \\ \vdots & \ddots & & & & \\ 0 & & & & & \\ (-1)^k d_2 \cdots d_{k+1} & \ddots & \ddots & & & \\ & \ddots & \ddots & & & \\ 0 & & -(-1)^n d_{n-k+1} \cdots d_n & 0 & \ddots & 0 \end{pmatrix}.$$

4. Numerical examples. In this section we consider the convergence for banded and block banded matrices, comparing the classical and the recursive Jacobi method with different ISAI preconditioners. Table 4.1 to 4.7 show the convergence behavior for different iterative solvers.

In Algorithm 5 we display an OpenMP code for the bidiagonal recursive solver. Table 4.8 shows the run times for this code on an Intel Xeon 'Skylake'. On this hardware, the recursive code is a factor 2.4 faster compared to the sequential bidiagonal solver.

5. A new hybrid preconditioner. The numerical examples show some weakness of ISAI and SAI preconditioners for triangular matrices. ISAI has the advantage that the main diagonal entries of the iteration matrix L_0 are zero which - theoretically - guarantees convergence. SAI has the advantage that it minimizes L_0 in the

k/n	100	200	400	800	1600
Jac. $k=0$	99	149	157	161	165
rec.Jac. $k=0$	7	8	8	8	8
Jac. $k=1$	49	74	78	80	82
rec.Jac. $k=1$	6	7	7	7	7
Jac. $k=2$	33	49	52	53	55
rec.Jac. $k=2$	6	6	6	6	6
Jac. $k=3$	24	37	39	40	41
rec.Jac. $k=3$	5	6	6	6	6

TABLE 4.3

Iteration count for preconditioned Jacobi solvers with different preconditioner for $L = tridiag(.9, 1, 0)$, *rhs* $ones(n)/\sqrt{n}$, *relative error* 10^{-6}. *The used preconditioner are ISAI to pattern* $abs(L)^k$.

k	E	$kron(T,T)$	$kron(T^2,T^2)$	$kron(T^4,T^4)$	$kron(T^7,T^7)$	$kron(T^{14},T^{14})$
Jac.	78	36	22	12	7	3
rec.Jac.	7	6	5	4	3	2

TABLE 4.4

Iteration count for preconditioned Jacobi solvers with different ISAI preconditioner for $n = 40*40$ $T = tridiag(1, 1, 0)$ *and* $L = kron(T, E) + kron(E, T)$, *rhs* $ones(n)/\sqrt{n}$, *relative error* 10^{-6}. *The pattern of the ISAI preconditioner is given by* $kron(T^k, T^k)$.

k	$kron(E,E)$	$kron(T,T)$	$kron(T^2,T^2)$
Jac.	38	8	4
rec.Jac.	9	4	3

TABLE 4.5

Iteration count for preconditioned Jacobi solvers with different ISAI preconditioner for $n = 20*20$ $T = tridiag(1, 0.9, 0.9^2, 0.9^3)$ *and* $L = kron(T, E) + kron(E, T)$, *rhs* $ones(n)/\sqrt{n}$, *relative error* 10^{-9}.

it2	it1=1	it1=2	it1=3	it1=4	it1=5	it1=6	it1=7	it1=8
$k=0$	*	*	*	*	*	*	*	*
$k=1$	166	83	41	20	10	5	2	1
$k=2$	83	41	20	10	5	2	1	0
$k=3$	*	*	*	*	*	*	*	*

TABLE 4.6

Iteration count for preconditioned Jacobi solvers with different preconditioner for $L = pentdiag(0.99^2, 0.99, 1, 0, 0)$, *rhs* $ones(n)/\sqrt{n}$, *relative error* 10^{-6}, $n = 1000$. *The used preconditioner are ISAI to pattern* $abs(L)^k$. *The iteration uses it1 steps of recursive Jacobi and then does it2 steps of Jacobi based on the last* L_0.

it2	it1=1	it1=2	it1=3	it1=4	it1=5	it1=6	it1=7
$k=0$	*	*	*	*	*	*	*
$k=1$	21	10	5	2	1	0	0
$k=2$	10	5	2	1	0	0	0
$k=3$	63	31	15	7	3	1	0
$k=4$	7	3	1	0	0	0	0
$k=5$	5	2	1	0	0	0	0

TABLE 4.7

Iteration count for preconditioned Jacobi solvers with different preconditioner for $L = pentdiag(0.9^2, 0.9, 1, 0, 0)$, *rhs* $ones(n)/\sqrt{n}$, *relative error* 10^{-6}, $n = 1000$. *The used preconditioner are ISAI to pattern* $abs(L)^k$. *The iteration uses it1 steps of recursive Jacobi and then does it2 steps of Jacobi based on the last* L_0.

Algorithm 5 Accelerated unpreconditioned Jacobi method for bidiagonal L in OpenMP.

Initialize d subdiagonal entries and $x = b$ rhs;
$kk = 1$;
for $k = 1 : log2(n)$ **do**
  $nk = n - kk$;
  ♯pragma omp parallel for schedule(static) private(jk)
  for $j = 0 : nk - 1$ **do**
    $jk = kk + j$; $f[j] = d[j] * x[j]$; $dd[jk] = d[jk]$;
  end for
  ♯pragma omp parallel for schedule(static) private(jk)
  for $j = 0 : nk - 1$ **do**
    $jk = kk + j$; $x[jk] += f[j]$; $d[j] *= dd[jk]$;
  end for
  $kk* = 2$;
end for

threads k	runtime
$k = 1$ direct sequential	2846532
$k = 1$ iterative	14.953.114
2	7.207.222
3	4.928.487
4	3.684.383
5	3.051.749
6	2.571.653
7	2.264.778
8	1.995.100
9	1.884.611
10	1.692.556
11	1.678.860
12	1.541.541
13	1.552.831
14	1.447.076
15	1.467.833
16	1.378.259
17	1.407.940
18	1.330.104
19	1.355.592
20	1.297.212
22	1.263.795
24	1.241.762
26	1.231.044
28	1.218.802
30	1.215.787
32	1.202.264
34	1.190.823
36	1.187.775
38	1.185.031
40	1.183.345

TABLE 4.8

Parallel runtimes for different number of threads in OpenMP. The best speedup relative to the sequential solver is therefore given by 2.4. The best speedup compared to the iterative solver with 1 thread is given by 12.6. The code was run for the bidiagonl matrix with subdiagonal entries all 0.5, size $n = 10^9$, rhs $b = ones/\sqrt{10^9}$ on the SuperMUC-NG of the LRZ Garching on an Intel Xeon 'Skylake' with 48 cores bundled in 8 domains (islands). The numerical results were done by Carsten Uphoff.

k/n	100	200	400	800	1600	3200
$k0 = 1, k1 = 1$	6	7	8	9	10	10
$k0 = 1, k1 = 2$	5	6	7	8	9	10
$k0 = 1, k1 = 3$	5	6	7	8	9	10
$k0 = 2, k1 = 2$	5	6	7	8	9	10
$k0 = 2, k1 = 3$	5	6	7	8	9	10
$k0 = 3, k1 = 3$	4	5	*	*	*	*

TABLE 5.1

Iteration count for preconditioned recursive Jacobi solvers with different preconditioner for $L = pentdiag(1, 1, 1, 0, 0)$*, rhs* $ones(n)/\sqrt{n}$*, relative error* 10^{-6}*. The used preconditioner are MISAI to pattern* $abs(L)^{k0}$ *and* $abs(L)^1$*.*

k/n	100	200	400	800	1600	3200
$k0 = 1, k1 = 1$	6	6	6	6	6	6
$k0 = 1, k1 = 2$	5	5	5	5	5	5
$k0 = 1, k1 = 3$	5	5	5	4	4	4
$k0 = 2, k1 = 2$	5	5	5	5	5	5
$k0 = 2, k1 = 3$	4	4	4	4	4	4
$k0 = 3, k1 = 3$	4	5	6	7	8	8

TABLE 5.2

Iteration count for preconditioned recursive Jacobi solvers with different preconditioner for $L = pentdiag(0.9^2, 0.9, 1, 0, 0)$*, rhs* $ones(n)/\sqrt{n}$*, relative error* 10^{-6}*. The used preconditioner are MISAI to pattern* $abs(L)^{k0}$ *and* $abs(L)^1$*.*

Frobenius norm. But for convergence $||L_0||_2$ should be less than 1. Therefore - as a compromise - we want to define a preconditioner that combines the desired properties: main diagonal of L_0 all zero, and $||L_0||$ small. To this end in a first step relative to a pattern, e.g. $S_0 = abs(L)^{k0}$, the ISAI preconditioner M_0 is determined. Now we allow a larger pattern, e.g. $S_1 = abs(L)^{k1}$. Relative to the difference pattern $S_2 = S_1 \backslash S_0$ we consider the MSPAI minimization

$$\min_{S_2} \|L(M_0 + M_1) - E\|_F = \min_{S_2} \|LM_1 - (E - LM_0)\|_F = \min_{S_2} \|LM_1 - B\|_F$$

combining the overall preconditioner via $M := M_0 + M_1$. Then it holds

$$L_0 = LM - E = L(M_0 + M_1) - E = LM_1 - (E - LM_0)$$

where $E - LM_0$ is zero on pattern S_0, and $LM_1 = (E + \tilde{L})M_1$ is zero above the pattern of S_2, hence also for S_0. Furthermore, the new iteration matrix L_0 minimizes at least the Frobenius norm. This minimization problem can be solved efficiently in parallel according to the MSPAI method [17]. Furthermore, in many cases entries of M_0 are directly given as the negative entries of L, compare theorem 2.3. Therefore, in the following we will call this preconditioner MISAI, relativ to pattern S_0 and S_1. Note, that such a preconditioner might be useful in general for improving a given preconditioner, e.g. block Jacobi peconditioner.

The tables 5.1 and 5.2 show a stabilizing effect of the MISAI preconditioner by means of the additional norm minimization.

6. Conclusions. In this paper we have shown that in many cases ISAI preconditioned and recursive accelerated Jacobi iterations represent a valuable solution method for sparse triangular matrices esp. in a parallel environment. The examples show that ISAI preconditioner reduces the number of iterations significantly. Furthermore, the recursive form of the Jacobi method leads to fast convergence and less operations, esp. of the patterns of the powers of the iteration matrix L_0 remains sparse. A hybrid preconditioner MISAI has also be introduced for improving an ISAI or block Jacobi method by an additional MSPAI norm minimization step.

Note that the developed solution methods can also be applied on twisted bidiagonal matrices that are used in eigenvalue solvers [23].

The next necessary step is a parallel implementation of the presented methods.

REFERENCES

[1] F. L. Alvarado, R. Schreiber, *Optimal parallel solution of sparse triangular systems*, SIAM J. Sci. Comput., 14 (1993), pp. 446–460.

[2] C. Anderson, Y. Saad, *Solving sparse triangular systems on parallel computers*, Intl. J. High Speed Comput., 1 (1989), pp. 73–96.

[3] H. Anzt, J. Bräckle, J. Dongarra, T. Huckle, *Incomplete Sparse Approximate Inverses for Parallel Preconditioning*, Parallel Computing 71, pp. 1-22, 2018.

[4] O. Axelsson, V. Eijkhout, *Vectorizable preconditioners for elliptic difference equations in three space dimensions*, J. Comput. Appl. Math., 27 (1989), pp. 299-321.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, Ch. Romine, H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 2nd ed., 1994.

[6] M. W. Benson, *Iterative solution of large scale linear systems*, Master's thesis, Lakehead University, Thunder Bay, ON, 1973.

[7] M. W. Benson, P. O. Frederickson, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math., 22 (1982), pp. 127–140.

[8] T. Chan, *A vectorizable variant of some ICCG methods*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 350-356.

[9] E. Chow, A. Patel, *Fine-grained parallel incomplete LU factorization*, SIAM J. Sci. Comput., 37 (2015), pp. C169–C193.

[10] E. Chow, Y. Saad, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[11] E. Chow, *Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns* , Int. J. High Perf. Comput. Appl, 15 (2001), pp. 56-74.

[12] M. J. Grote, T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[13] S. W. Hammond, R. Schreiber, *Efficient ICCG on a shared memory multiprocessor*, Intl. J. High Speed Comput., 4 (1992), pp. 1–21.

[14] N. J. Higham, *Stability of parallel triangular system solvers*, SIAM Journal on Scientific Computing, 16, (1995).

[15] N. J. Higham, A. Pothen, *Stability of the partitioned inverse method for parallel solution of sparse triangular systems*, SIAM Journal on Scientific Computing, 15, (1994).

[16] T. Huckle, *Parallel Approximate LU Factorizations for Sparse Matrices*, preprint 2019.

[17] T. Huckle., A. Kallischko, *Frobenius Norm Minimization and Probing for Preconditioning*, in International Journal of Computer Mathematics 84(8) 1225-1248, 2007 (special issue on Fast Iterative and Preconditioning Methods for Linear and Nonlinear Systems)

[18] J. Mayer, *Parallel algorithms for solving linear systems with sparse triangular matrices*, Computing, 86 (2009), pp. 291–312.

[19] J. A. Meijerink, 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 (1977), pp. 148162.

[20] M. Naumov, *Parallel solution of sparse triangular linear systems in the preconditioned iterative methods on the GPU*, Tech. Report NVR-2011-001, NVIDIA, 2011.

[21] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2nd ed., 2003.

[22] J. H. Saltz, *Aggregation methods for solving sparse triangular systems on multiprocessors*, SIAM J. Sci. Comput., 11 (1990), pp. 123–144.

[23] P. R. Willems, B. Lang, *Twisted factorizations and qd-type transformations for the MR3 algorithm-new representations and analysis*, SIAM J. Matrix Anal. Appl., 33(2):523-553, 2012.

[24] M. M. Wolf, M. A. Heroux, E. G. Boman, *Factors impacting performance of multithreaded sparse triangular solve*, High Performance Computing in Computational Science, VECPAR

2010, 6449 (2010), pp. 32–44.