INCOMPLETE SPARSE APPROXIMATIONS OF MATRICES, INVERSES OF MATRICES, AND THEIR FACTORIZATIONS

JÜRGEN BRÄCKLE * AND THOMAS K. HUCKLE †

Abstract. The paper describes relationships between the recently introduced iterative construction of Incomplete LU factorizations and Sparse Approximate Inverses derived via Frobenius norm minimization. Furthermore, the application of ILU preconditioners needs an efficient parallel solver for sparse triangular linear systems. Therefore, various preconditioners for sparse triangular matrices are derived and compared. Especially, the Incomplete Sparse Approximate Inverse (ISAI) matrix that is closely related to the Diagonal Block Approximate Inverse (DBAI) of Benson can be seen as a generalization of the Jacobi method leading to fast convergence.

Key words. Incomplete LU factorization, Sparse Approximate Inverses, preconditioning, Jacobi method, sparse triangular linear system, parallel computing

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

1. Introduction. Solving linear systems of equations $Ax = b$ for sparse matrix A iteratively via the method of conjugate gradients or GMRES usually needs a preconditioner P or M in the form $P^{-1}Ax = P^{-1}b$ or $MAx = Mb$ in order to get satisfactorily convergence. Especially on a highly parallel computing architecture the preconditioner should additionally be easy to derive and to apply. The family of direct preconditioners like Gauss-Seidel (GS) or Incomplete LU decomposition (ILU) often leads to improved convergence but is strongly sequential [16, 3, 14]. On the other side preconditioners like Sparse Approximate Inverses (SAI) M minimizing $||AM - I||$ in the Frobenius norm show good parallel performance but lead to unsatisfactorily convergence [4, 5, 12, 9, 7, 8, 11, 18].

Furthermore, for ill-conditioned A also the preconditioner should be ill-conditioned - otherwise it will not improve the conditioning, resp. the convergence. For ILU the ill-conditioning is no problem if the occurring triangular linear systems are solved directly. But to take easy advantage of parallel architectures iterative methods can be considered which makes it necessary to define efficient parallel preconditioners for triangular matrices. This leads to (Block) Jacobi methods and Sparse Approximate Inverse preconditioners. But, approximating the inverse suffers from the fact that the inverse of a sparse matrix is usually nearly dense which hampers the quality of a sparse approximation.

In recent work E. Chow and A. Patel introduced new ideas to reconcile the drawbacks of preconditioners, namely slow convergence and bad parallelization [6]. The underlying idea is to use an iterative, but inherently parallel process to compute ILU. The arising triangular linear systems can then also be solved iteratively (by Jacobi-like iterations) to avoid the sequential character of triangular solves.

In this paper we show the intimate connection between Incomplete (ILU-like) preconditioners and sparse approximations based on the minimization of the Frobenius norm. This leads to new classes of sparse approximate (inverse) factorized preconditioners, and to various iterative procedures to derive ILU and related preconditioners efficiently in parallel.

* Department of Informatics, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Munich, Germany (braeckle@in.tum.de).

† Department of Informatics, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Munich, Germany (huckle@in.tum.de).

Furthermore, we tackle the still open problem of solving ill-conditioned triangular systems iteratively. Instead of well known parallel direct solvers for triangular systems based on level scheduling or sparse triangular factorizations [1, 2, 10, 13, 15, 17, 20], we use stationary iterative methods. Here, also incomplete preconditioners closely related to ILU, SAI and Diagonal Block Approximate Inverses (DBAI) [4, 5] are introduced. It is shown that the stationary iterations derived by these preconditioners are generalizations of the Jacobi method and allow fast convergence. Numerical examples show the usefulness of the new methods.

1.1. Incomplete LU Decomposition. The Incomplete LU factorization can be seen as a curtailed Gaussian Elimination where the computations are restricted to a certain sparsity pattern, e.g. the original sparsity pattern $\mathcal{S}(A)$ of A. Hence, L, U, and A are reduced to the pattern $\mathcal{S}$, satisfying

$$(LU)_{i,j} = A_{i,j} \quad \text{for } (i,j) \in \mathcal{S} \,. \tag{1.1}$$

The usual way of computing the incomplete factors L and U is sequential and given by a reduced version of the Gaussian elimination process:

Algorithm 1 Conventional ILU

```
for i = 2 : n do
  for k = 1 : i − 1 and (i, k) ∈ S do
    a_{i,k} = a_{i,k} / a_{k,k}
    for j = k + 1 : n and (i, j) ∈ S do
      a_{i,j} = a_{i,j} − a_{i,k} a_{k,j}
    end for
  end for
end for
```

Generalizations allow a thicker a priori pattern like the pattern of $|A|^k$ or use a threshold in the Gaussian elimination replacing small entries by zero. In the modified ILU approach (MILU) the deleted/ignored entries are not deleted but moved to the main diagonal obtaining the sparse pattern and the column sum, resp. the action of the preconditioner on the all-ones vector [16, 3].

In this form the computation of the factors L and U is strictly sequential, and also solving the resulting triangular systems in L and U in the preconditioning step is strictly sequential. Therefore, Chow and Patel suggested in [6] a parallel iterative way of computing ILU. They consider (1.1) as a set of nonlinear equations

$$(LU - A)_{i,j} = 0 \quad for\ (i,j) \in \mathcal{S} \,, \quad \mathcal{S}(L) \cup \mathcal{S}(U) \subseteq \mathcal{S} \tag{1.2}$$

or

$$\sum_{k=1}^{\min(i,j)} l_{i,k} u_{k,j} = a_{i,j}, \quad (i,j) \in \mathcal{S}, \quad \mathcal{S}(L) \cup \mathcal{S}(U) \subseteq \mathcal{S} \,. \tag{1.3}$$

Thereby, all entries of L and U outside the given pattern $\mathcal{S}$ are equal to zero, and the equations derive only from entries $(i,j) \in \mathcal{S}$. In this form the solution method for (1.3) heavily depends on the ordering of this enlarged system of $|\mathcal{S}|$ bilinear equations. Choosing the ordering according to the Gaussian elimination results in a sequential

direct solver. But to allow parallelism we can consider the iteration freezing L (with some initial guess) and using (1.3) to solve for U, and vice versa. Repeating this step generates an iteration that converges against ILU. Instead of solving the resulting linear systems directly a Jacobi like iteration is applied always using one equation independently in parallel to update an entry in L or U via

$$l_{i,j} = \frac{1}{u_{j,j}}(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k}u_{k,j}) \tag{1.4}$$

$$u_{i,j} = \frac{1}{l_{i,i}}(a_{i,j} - \sum_{k=1}^{i-1} l_{i,k}u_{k,j}) \ . \tag{1.5}$$

Thereby, we always choose one equation (out of the $|\mathcal{S}|$ equations in (1.3)) containing the term $l_{i,j}u_{j,j}$ for updating $l_{i,j}$, and the equation containing $l_{i,i}u_{i,j}$ for updating $u_{i,j}$ based on the old L and U. In this form all updates are independent and can be done fully in parallel. Furthermore, locally convergence against ILU is proven in [6].

Like in the Gauss-Seidel approach the convergence can be improved (without loosing parallel efficiency) by inserting newer already updated values in a restricted fashion (based on some coloring or distribution of the matrices on different processors). The fixed point iteration introduced by Chow and Patel has the form

Algorithm 2 Iterative ILU

```
Set unknowns l_{i,j} and u_{i,j} to initial values
for sweep = 1,2,... until convergence do
  parallel for (i,j) ∈ S do
    if i > j then
      l_{i,j} = a_{i,j} − (1/u_{i,j}) Σ_{k=1}^{j−1} l_{i,k} u_{k,j}
    else
      u_{i,j} = a_{i,j} − Σ_{k=1}^{i−1} l_{i,k} u_{k,j}
    end if
  end parallel for
end for
```

After constructing approximations to the ILU factors in this way, in each iteration step of the original given linear equations in A the preconditioners have to be solved, e.g. in the form $Ly = c$. Here, in [6] stationary Jacobi iteration is suggested for overcoming the sequential nature of triangular solves. But note that either the computed ILU factors are well-conditioned leading to unsatisfactory convergence in the original problem in A but possibly fast convergence for solving $Ly = c$. Or, on the other side, ill-conditioned LU factors are necessary for fast convergence in $Ax = b$, but will lead to unsatisfactorily convergence in the stationary iterations for solving $Ly = c$. Hence, efficient iterative triangular solves remain an open problem in a parallel environment.

For parallel direct solvers different strategies have been developed (see [6], esp. [1, 2, 10, 13, 15, 17, 20]). But here we will only discuss iterative solvers.

1.2. Sparse Approximate Inverses. A fully parallel but similar approach was introduced by Kolotilina and Yeremin in [12] considering Sparse Approximate Inverses of A again of certain nonzero pattern $\mathcal{S}$

$$\min_{G \in \mathcal{S}} \|GA - I\|_W = \min_{G \in \mathcal{S}} \text{trace}((GA - I)W(GA - I)^T) \ .$$

For symmetric positive definite A we can choose $W = A^{-1}$ resulting in

$$(GA)_{i,j} = I_{i,j}, \quad (i,j) \in \mathcal{S}$$

with $G_{i,j} = 0$ for $(i,j) \notin \mathcal{S}$. Assuming that $\mathcal{J}$ is given by the pattern of column A_j, this leads for the i-th row of G to the equation:

$$G(i,\mathcal{J})A(\mathcal{J},\mathcal{J}) = I(i,\mathcal{J}) \ .$$

In the factorized case similarly one derives

$$(G_L A)_{i,j} = (L_A)_{i,j} \ for \ (i,j) \in \mathcal{S}_L$$

with Cholesky decomposition $A = L_A L_A^T$ and $A^{-1} \approx G_L^T G_L$ (see [12]).

For general A the choice of $W = I$ leads to the so called Sparse Approximate Inverses (SAI) minimizing in the Frobenius with given pattern $\mathcal{S}$

$$\min_{M \in \mathcal{S}} \|AM - I\|_F^2 = \sum_{j=1}^{n} \min_{M_j \in \mathcal{S}} \|AM_j - I_j\|_2^2 \tag{1.6}$$

where M_j and I_j denote the j-th column. Hence, we can consider all columns independently $\min ||AM_j - I_j||_2$, and the sparsity patterns in A and M allow a simplification of the resulting Least Squares problems on submatrices of A in the form

$$\min_{M_j(\mathcal{J})} \|A(\mathcal{I},\mathcal{J})M_j(\mathcal{J}) - I_j(\mathcal{I})\|_2^2 \ , \tag{1.7}$$

where $\mathcal{J}$ is again the allowed pattern in the j-th column of M and $\mathcal{I}$ is the so called shadow of $\mathcal{J}$, indicating the nonzero rows in $A(:,\mathcal{J})$.

The preconditioner (1.6) was generalized in [19] to target matrices B different from I

$$\min_{M \in \mathcal{S}} \|AM - B\|_F^2 \ .$$

Furthermore, in [11] this was extended to include also probing in the form

$$\min_{M} \| \begin{pmatrix} A \\ \rho F \end{pmatrix} M - \begin{pmatrix} I \\ \rho E \end{pmatrix} \|_F^2 \tag{1.8}$$

where - for any chosen collection E of probing vectors - F is given by $F = EA$ and $\rho \geq 0$ is a weighting factor. So the preconditioner is forced to match the behaviour of the inverse of A on the probing vectors in a better way steered by the weight ρ. Possible choice is - similar to MILU - $E = (1,1,...,1)^T$. In [11] then also an iterative scheme for computing a sparse approximate LU decomposition has been introduced for $LU \approx A$ and $LAU \approx I$, by solving (1.8) e.g. in the form

$$\min_{U^{(s)}} \|L^{(s-1)} \cdot U^{(s)} - A\|_F \ , \ \min_{L^{(s)}} \|{U^{(s)}}^T \cdot {L^{(s)}}^T - A^T\|_F \ , \ s = 1,2,... \ , \tag{1.9}$$

starting from an initial guess $L^{(0)}$. In the same way, there were defined approximate inverse factorizations via the iteration

$$\min_{U^{(s)}} \|(L^{(s-1)}A) \cdot U^{(s)} - I\|_F \ , \ \min_{L^{(s)}} \|({U^{(s)}}^T A^T) \cdot {L^{(s)}}^T - I\|_F \ , \ s = 1,2,... \ , \tag{1.10}$$

always freezing matrix L and solving for U, and vice versa. In this way an iterative procedure has been introduced for deriving sparse approximations for A, A^{-1}, and their factorizations that allow to include additional probing conditions (1.8). Such iterative schemes are related to Alternating Linear Systems (ALS) methods where always one part is fixed and the optimization/solution is done only for the remaining unknowns; repeating this procedure in a cyclic manner results in a sequence of L and U that converges locally against the ILU factorization we are looking for.

2. Incomplete Sparse Approximate Preconditioners.

2.1. Incomplete Sparse Approximate LU Factorization. First let us compare the linear systems for computing columns of a preconditioner in the ILU approach and the SAI method. Therefore, let us start with the special case that we want to compute a sparse factor U for an approximate LU decomposition for given L and given triangular pattern. In SAI minimizing the Frobenius norm

$$\min_U \|LU - A\|_F$$

leads to the Least Squares problems

$$L^T(\mathcal{I},\mathcal{J})L(\mathcal{I},\mathcal{J}) \cdot U(\mathcal{J},j) = L^T(\mathcal{I},\mathcal{J})A(\mathcal{I},j) \tag{2.1}$$

minimizing

$$\|L(\mathcal{I},\mathcal{J}) \cdot U(\mathcal{J},j) - A(\mathcal{I},j)\|_2 \ .$$

Because the diagonal entries of L and U are nonzero, the index set $\mathcal{J}$ is contained in $\mathcal{I}$, and therefore the rectangular matrix $L(\mathcal{I},\mathcal{J})$ can be written as two blocks

$$L(\mathcal{I},\mathcal{J}) = \begin{pmatrix} L(\mathcal{J},\mathcal{J}) \\ L(\mathcal{I}\setminus\mathcal{J},\mathcal{J}) \end{pmatrix}$$

with the lower triangular square matrix $L(\mathcal{J},\mathcal{J})$. So $L(\mathcal{J},\mathcal{J})$ is regular and we can also consider a shorter way to define an U-factor in an incomplete scheme derived by solving

$$L(\mathcal{J},\mathcal{J}) \cdot U(\mathcal{J},j) = A(\mathcal{J},j) \ , \tag{2.2}$$

instead of (2.1). But solving (2.2) is exactly what is done in the iterative scheme (1.3), but with computing the entries of one column sequentially, and all columns independently in parallel. So generalizing the SAI and the ILU approach we end up with factor U by solving

$$\min_{U(\mathcal{J},j)} \| \begin{pmatrix} L(\mathcal{J},\mathcal{J}) \\ \rho L(\mathcal{I}\setminus\mathcal{J},\mathcal{J}) \end{pmatrix} U(\mathcal{J},j) - \begin{pmatrix} A(\mathcal{J},j) \\ \rho A(\mathcal{I}\setminus\mathcal{J},j) \end{pmatrix} \|_2 \ .$$

For $\rho = 0$ this results in a certain update sequence for the iterative computation of the ILU factorization. Repeating the same procedure with chosen U for updating L we get local convergence against the exact ILU. For $\rho = 1$ we get a similar method resulting in another approximate LU decomposition that is minimizing the Frobenius norm.

Compared with (1.3) and the iterative ILU algorithm, the SAI-like iteration for computing the ILU with $\rho = 0$ has the advantage that it is also fully parallel but always uses new updated values per step because inside a column the new entries are computed sequentially.

2.2. Incomplete Sparse Approximate Inverse LU Factorization. The same iterative process can be derived for Incomplete Inverse LU decomposition. Therefore, we assume again fixed L and recompute U with chosen pattern $\mathcal{S}$ via

$$\min_{U \in \mathcal{S}} \|(LA)U - I\|_F \ .$$

This leads with $B := LA$ to the Least Squares problems

$$B^T(\mathcal{I}, \mathcal{J})B(\mathcal{I}, \mathcal{J}) \cdot U(\mathcal{J}, j) = B^T(\mathcal{I}, j)$$

minimizing

$$\|L(\mathcal{I}, :)A(:, \mathcal{J}) \cdot U(\mathcal{J}, j) - I(\mathcal{I}, j)\|_2 \ .$$

Here, $\mathcal{I}$ again denotes the nonzero rows of $(LA)(:, \mathcal{J}) = B(:, \mathcal{J})$. Repeating the same minimization for computing L for given U, gives a SAI-like iteration to compute an approximate LU decomposition of the inverse of A. If the pattern $\mathcal{S}$ also contains the diagonal entries then $\mathcal{J} \subseteq \mathcal{I}$. Furthermore, if $B(\mathcal{J}, \mathcal{J})$ is nonsingular - which will be satisfied if there exists a factorization $A = LU$ and the iterates are close to this exact LU factorization - we can simplify this to

$$\min \|L(\mathcal{J}, :)A(:, \mathcal{J}) \cdot U(\mathcal{J}, j) - I(\mathcal{J}, j)\|_2 = \min \|B(\mathcal{J}, \mathcal{J}) \cdot U(\mathcal{J}, j) - I(\mathcal{J}, j)\|_2 \ .$$

As a regularization for singular or ill-conditioned $B(\mathcal{J}, \mathcal{J})$ we can again for U include the non-square part with weight ρ in the form

$$\min_{U(\mathcal{J},j)} \| \begin{pmatrix} B(\mathcal{J}, \mathcal{J}) \\ \rho B(\mathcal{I} \setminus \mathcal{J}, \mathcal{J}) \end{pmatrix} U(\mathcal{J}, j) - \begin{pmatrix} I(\mathcal{J}, j) \\ \rho I(\mathcal{I} \setminus \mathcal{J}, j) \end{pmatrix} \|_2 \ .$$

Furthermore, again probing conditions can be added to the minimization in the form

$$\min_{U(\mathcal{J},j)} \| \begin{pmatrix} B(\mathcal{J}, \mathcal{J}) \\ \rho B(\mathcal{I} \setminus \mathcal{J}, \mathcal{J}) \\ \sigma(v^T B)(\mathcal{I}, \mathcal{J}) \end{pmatrix} U(\mathcal{J}, j) - \begin{pmatrix} I(\mathcal{J}, j) \\ \rho I(\mathcal{I} \setminus \mathcal{J}, j) \\ \sigma v^T(j) \end{pmatrix} \|_2$$

with given probing vector v and weight σ.

To derive a new preconditioning scheme, in a similar approach we can mix direct and inverse factors in the form

$$\min_U \|LA - U\|_F \ , \tag{2.3}$$

where L is an approximate factor of the inverse and U is a given approximate factor of A, e.g. the upper triangular part of A. This has the advantage that in a parallel environment for each preconditioning step only one triangular system has to be solved and not two.

Similar to (2.3) we can consider a given explicit factor L of A to compute an inverse factor U via

$$\min_U \|AU - L\|_F \ . \tag{2.4}$$

2.3. Incomplete Sparse Approximate Inverses. The above approach for mixing incomplete and sparse approximate triangular factorizations suggests to also extend the original SAI method

$$\min_{M(\mathcal{J},j)} \|A(\mathcal{I},\mathcal{J})M(\mathcal{J},j) - I(\mathcal{I},j)\|_2$$

to an incomplete version

$$\min_{M(\mathcal{J},j)} \|A(\mathcal{J},\mathcal{J})M(\mathcal{J},j) - I(\mathcal{J},j)\|_2$$

or

$$A(\mathcal{J},\mathcal{J})M(\mathcal{J},j) = I(\mathcal{J},j)$$

which is well defined only for $\mathcal{J} \subseteq \mathcal{I}$ and nonsingular $A(\mathcal{J},\mathcal{J})$. The property $\mathcal{J} \subseteq \mathcal{I}$ is satisfied if the diagonal entries of A are nonzero and we choose pattern set given by $|A|^k$. The property that $A(\mathcal{J},\mathcal{J})$ is nonsingular is satisfied for triangular A or positive definite A.

Let us assume that the pattern of the preconditioner is given by $|A|^k$, for some integer k, and that $A(\mathcal{J},\mathcal{J})$ are well-defined nonsingular. Then we can describe the incomplete approximation equivalently by the condition

$$(AM - I).*(|A|^k) = 0 \quad for \quad \mathcal{S}(M) = \mathcal{S}(|A|^k) \ .$$

Here, $.*$ denote the Hadamard product. For $k = 0$ this leads to the $M = diag(A)^{-1}$ and thus to the Jacobi preconditioner. In general for $k > 0$ the preconditioner leads to zero blocks of $AM - I$ in the prescribed pattern of $|A|^k$.

A first advantage of this Incomplete SAI is a reduction of computational costs because solving a linear system in sparse $A(\mathcal{J},\mathcal{J})$ will be cheaper than solving a least squares problem in $A(\mathcal{I},\mathcal{J})$. Furthermore, in many cases the incomplete preconditioner leads to faster convergence as shown in section 4. This Incomplete Sparse Approximate Inverse (ISAI) preconditioner was introduced by Benson in [4, 5] as Diagonal Block Approximate Inverse (DBAI). In this form the DBAI preconditioner was also considered in [7].

A weighted, regularized version including probing can also be defined for general A in the form

$$\min \| \begin{pmatrix} A(\mathcal{J},\mathcal{J}) \\ \rho A(\mathcal{I}\setminus\mathcal{J},\mathcal{J}) \\ \sigma(v^T A)(\mathcal{J}) \end{pmatrix} M(\mathcal{J},j) - \begin{pmatrix} I(\mathcal{J},j) \\ \rho I(\mathcal{I}\setminus\mathcal{J},j) \\ \sigma v(j) \end{pmatrix} \|_2 \ .$$

Here, we will consider this preconditioner only for triangular matrices. The general case will be topic of an upcoming paper.

REMARK 1 (Application to Symmetric Positive Definite Matrices). *In many cases we are looking for Cholesky factorization of symmetric positive definite A. We can modify the iteration scheme to enforce symmetry e.g.*

$$\min_{U_{new}} \|U_{old}^T U_{new} - A\|_F \ , \quad U := (U_{old} + U_{new})/2 \ .$$

The same symmetrization can be applied for incomplete versions or for inverse factorization, but not to the mixed version (2.3) or (2.4).

REMARK 2 (Matrix free implementation). *In many cases the underlying matrices have special structure, e.g. constant coefficients along diagonals. Then most of the linear systems that have to be solved for constructing the preconditioner are identical, and thus the setup is practically for free. Furthermore, often there is no explicit representation of the given matrix. In such cases the application of the preconditioner Mx can also be formulated in a matrix free form*

$$Mx = \sum_{j=1}^{n} x_j M_j = \sum_{j=1}^{n} x_j (A(\mathcal{J}_j, \mathcal{J}_j)^{-1} e_j(\mathcal{J}_j)) \ .$$

Then, we build Mx columnwise, and each column M_k is always computed on demand.

REMARK 3 (Approximate Sparse Triangular Solves). *The relationship between the triangular ISAI preconditioner and the iterative ILU (Algorithm 2) is also reflected by the similarity of the algorithms. Here, a possible implementation of ISAI for computing a triangular preconditioner M for triangular matrix L:*

Algorithm 3 ISAI for triangular L

```
for j = 1 : n do
  m_{i,j} = 1/l_{i,j}
  for k = j + 1 : n and k ∈ S(l(:, j)) do
    m_{k,j} = 0
    for r = j : k − 1 and r ∈ S(l(:, j)) do
      m_{k,j} = m_{k,j} − l_{k,r} m_{r,j}
    end for
    m_{k,j} = m_{k,j} / l_{k,k}
  end for
end for
```

Note, that this algorithm can be seen as a collection of incomplete approximate triangular solves of the form $LM_j = e_j$.

REMARK 4 (Iterative Computation of modified ILU (MILU)). *We can combine the incomplete part and the probing part in the iterative ILU approach in the form*

$$\min_{U(\mathcal{J},j)} \| \begin{pmatrix} L(\mathcal{J},\mathcal{J}) \\ \rho(v^T L)(\mathcal{J}) \end{pmatrix} U(\mathcal{J}, j) - \begin{pmatrix} A(\mathcal{J}, j) \\ \rho(v^T A)(j) \end{pmatrix} \|_2 \ .$$

Then for each column of the preconditioner we have to solve a Least Squares problem where the matrix is a rank 1 extension of a triangular square block. Hence, the additional computational costs are negligible. By choosing $v = (1, ..., 1)^T$ and increasing ρ we can force the factorization to satisfy approximately $v^T(LU - A) \approx 0$ like in MILU.

3. Iterative Solution of Triangular Systems. After deriving an ILU decomposition it still remains the problem of solving ill-conditioned triangular systems $Ly = c$ efficiently in parallel. Besides the direct solution methods in [6] the stationary Jacobi method is recommended

$$y^{(s+1)} = y^{(s)} + D^{-1}(c - Ay^{(s)}) = D^{-1}c + D^{-1}(D - A)y^{(s)}$$

with $D = diag(A)$ and start $y^{(0)}$, e.g. $y^{(0)} = 0$.

To improve the convergence, only Block Jacobi or inverse preconditioners M_L are meaningful because of the triangular structure and the intended parallel implementation. Here, we consider the SAI and Incomplete SAI approach:

$$\min_{M_L(\mathcal{J},j)} \| \begin{pmatrix} L(\mathcal{J},\mathcal{J}) \\ \rho L(\mathcal{I}\setminus\mathcal{J},\mathcal{J}) \end{pmatrix} M_L(\mathcal{J},j) - \begin{pmatrix} I(\mathcal{J},j) \\ \rho I(\mathcal{I}\setminus\mathcal{J},j) \end{pmatrix} \|_2 \ .$$

The special case $\rho = 0$

$$\min_{M_L(\mathcal{J},j)} \|L(\mathcal{J},\mathcal{J})M_L(\mathcal{J},j) - I(\mathcal{J},j)\|_2 \ , \ L(\mathcal{J},\mathcal{J})M_L(\mathcal{J},j) = I(\mathcal{J},j) \ , \tag{3.1}$$

gives the ISAI preconditioner for triangular L.

Again we can also include probing conditions relative to a given probing vector v

$$\min_{M_L(\mathcal{J},j)} \| \begin{pmatrix} L(\mathcal{J},\mathcal{J}) \\ \rho L(\mathcal{I}\setminus\mathcal{J},\mathcal{J}) \\ \sigma(v^T L)(\mathcal{J}) \end{pmatrix} M_L(\mathcal{J},j) - \begin{pmatrix} I(\mathcal{J},j) \\ \rho I(\mathcal{I}\setminus\mathcal{J},j) \\ \sigma v(j) \end{pmatrix} \|_2 \ .$$

This allows an improved stationary iteration based on the Richardson splitting on $(LM_L + I - I)(M_L^{-1}x) = c$ of the form

$$y^{(s+1)} = y^{(s)} + M_L(c - Ly^{(s)}) \ . \tag{3.2}$$

To derive fast convergence and allowing higher computational costs, in M_L we should allow also more dense sparsity pattern. This can be achieved by choosing the pattern of $|L|^k$ for some $k > 1$ and by allowing also a block pattern melting neighboring index sets together and using the union of the pattern of these indices as allowed common pattern for all of them. Distributing the blocks over different processors introduces more overall computations but does not destroy the parallelism. Furthermore, the pattern $|L|^k$ does not grow as fast with increasing k as in the general case. In our numerical examples the effect of blocking did not improve the convergence significantly. Therefore, we don't use blocking in the following.

The ISAI preconditioner (3.1) with pattern $|L|^k$ has some favourable properties. First, it is well defined. Furthermore, like in the Jacobi case, the iteration matrix $LM_L - I$ has zeros on the diagonal, therefore spectral radius zero. Hence, the stationary iteration is guaranteed to converge:

THEOREM 3.1. *For Block Jacobi preconditioner and for ISAI preconditioner (2.3) with triangular pattern that includes the diagonal entries, the stationary iteration (3.2) leads to zero blocks on the diagonal, therefore to spectral radius 0, and hence to convergence.*

Because $LM_L - I$ has zeros on the whole pattern $|L|^k$, for $k \geq 1$ we can expect faster convergence. Also the block Jacobi method where we use a certain block diagonal part shares these properties (spectral radius zero, convergence), but ISAI generates the zeros on more important positions while block Jacobi is concentrated on main diagonal blocks. Therefore, we expect for ISAI faster convergence.

Applying ISAI for general, nontriangular problems, the iteration matrix $AM - I$ will also be zero on the main diagonal, but there is no direct connection to the spectral radius or convergence in this case. Heuristically, one could argue, that if the chosen pattern is large enough, the entries not covered by the pattern are not important and small, and thus the spectral radius will also be less than 1.

It. ILU	1	2	3	4	5
cond	36.42	35.26	35.23	35.225	35.225
$max(max(\|LU - L_eU_e\|))$	0.0026	$7.4e-5$	$2.2e-6$	$6.4e-8$	$1.9e-9$
It. SAI	1	2	3	4	5
cond	45.78	45.41	45.40	45.40	45.40
$\|\|LU - A\|\|_F$	1.3495	1.3490	1.3490	1.3490	1.3490

TABLE 4.1

Condition number and convergence of the iteration for iterative ILU and SAI. Here, L_e and U_e denote the exact ILU.

It. MILU, $\rho = 0$	1	2	3	4	5
cond	36.42	35.26	35.23	35.225	35.225
It. MILU, $\rho = 1$	1	2	3	4	5
cond	18.9	21.0	21.21	21.22	21.22
It. MILU, $\rho = 2$	1	2	3	4	5
cond	11.6	13.8	14.4	20.6	27.8
It. MILU, $\rho = 3$	1	2	3	4	5
cond	9.9	40.1	*	*	*

TABLE 4.2

*Condition number for iterative computation for modified ILU with different weight ρ. Here, * denotes failure caused by negative diagonal entries.*

The ISAI condition for triangular L

$$(LM_L - I).*|L|^k = 0\ ,\ \mathcal{S}(M_L) = \mathcal{S}(|L|^k)$$

is closely related to the ILU condition for triangular L and U

$$(LU - A).*(|A|^k) = 0 \ \ for\ \ \mathcal{S}(U) = triu(|A|^k)\ ,\ \mathcal{S}(L) = tril(|A|^k)\ .$$

4. Numerical Examples.

4.1. ILU and MILU. First, we want to compare different methods for deriving ILU decompositions: The exact direct computation, the iteration based on the ALS fixed point iteration in ISAI, and the ALS iteration based on SAI, all with pattern of A. As example A in table 4.1 we choose the 5-point stencil for the 2D Laplacian of size 30×30, the condition of the ILU preconditioned system ($A \approx L_eU_e$) is given by $cond(L_e^{-1}AU_e^{-1}) = 35.328$. Obviously, there is fast convergence against the optimal LU factors, and the improvement in the condition number is better for ILU. For the same problem, the MILU Incomplete Cholesky decomposition gives $cond(L^{-1}AL^{-T}) = 9.007$. Table 4.1 shows that by including the MILU-like probing conditions (see Remark 4) it is possible to achieve smaller condition numbers, but one has to choose the weight ρ not to large; otherwise negative diagonal entries can occur and the condition number gets worse.

4.2. Triangular Solves: A first Example. We consider lower triangular matrices related to the constant coefficient 1D and 2D Laplacian to stencils $[-1, 2, -1]$ and

$$\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}\ .$$

For the 1D case we therefore consider $L = tridiag(-1, 1, 0)$ with inverse

$$L^{-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 1 & \cdots & 1 & 1 \end{pmatrix} .$$

To compute the ISAI preconditioner with pattern L we have to solve the linear systems

$$L(\mathcal{J}, \mathcal{J})M_j(\mathcal{J}) = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} M_j(\mathcal{J}) = e_1(1 : |\mathcal{J}|) .$$

Therefore, the ISAI preconditioners of different level are then given by a banded matrix of ones, and this band matches the all-ones lower triangular part of L^{-1}

$$L = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & & & & \vdots \\ 1 & & \ddots & & & \vdots \\ 0 & \ddots & & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & \cdots & 1 \end{pmatrix} .$$

The structure is bidiagonal for pattern L, and with $k+1$ diagonals of ones for pattern $|L|^k$. Then the iteration matrix looks like

$$LM - I = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & & & & & \vdots \\ 0 & & \ddots & & & & \vdots \\ -1 & \ddots & & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \end{pmatrix}$$

with a larger zero band, resp. zero block diagonal part allowing faster convergence.

For the 2D case we consider the block matrix $L_2 = kron(L, I) + kron(I, L)$ with the 1D matrix L from above with first column given by $(2, -1, 0, ..., 0, -1, 0, ..., 0)^T$. To compute the ISAI preconditioner with pattern L we have to solve the linear systems (or subsystems)

$$L(\mathcal{J}, \mathcal{J})M_j(\mathcal{J}) = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 2 & 0 \\ -1 & 0 & 2 \end{pmatrix} M_j(\mathcal{J}) = e_1(1 : |\mathcal{J}|) .$$

In the table 4.2 we present the number of stationary iterations for solving a linear system with random right-hand-side to the relative residual 10^{-6} for Block Jacobi, SAI and ISAI preconditioners and different patterns $|L|^k$. In the Block Jacobi we

N	10	20	30	40	50	60	nnz
JAC	18 (100)	38 (400)	58 (900)	78 (1600)	98 (2500)	118 (3600)	N^2
BJAC2	14 (200)	29 (800)	44 (1800)	59 (3200)	74 (5000)	89 (7200)	$2N^2$
BJAC3	16 (298)	33 (1198)	39 (2700)	60 (4798)	76 (7498)	79 (10800)	$3N^2$
BJAC4	14 (400)	24 (1600)	42 (3600)	49 (6400)	69 (10000)	74 (14400)	$4N^2$
BJAC5	11 (500)	23 (2000)	35 (4500)	47 (8000)	59 (12500)	71 (18000)	$5N^2$
SAI1	25 (280)	44 (1160)	62 (2640)	80 (4720)	97 (7400)	114 (10680)	nnz
SAI2	18 (521)	31 (2241)	44 (5161)	56 (9281)	68 (14601)	79 (21121)	$1.9nnz$
SAI3	14 (805)	25 (3605)	34 (8405)	44 (15205)	53 (24005)	62 (34805)	$3.2nnz$
SAI4	12 (1115)	21 (5215)	29 (12315)	37 (22415)	44 (35515)	52 (51615)	$4.7nnz$
SAI5	11 (1435)	18 (7035)	25 (16835)	32 (30835)	38 (49035)	44 (71435)	$6.5nnz$
ISAI1	9 (280)	19 (1160)	29 (2640)	39 (4720)	49 (7400)	59 (10680)	nnz
ISAI2	6 (521)	13 (2241)	20 (5161)	26 (9281)	33 (14601)	40 (21121)	$1.9nnz$
ISAI3	5 (805)	10 (3605)	15 (8405)	20 (15205)	25 (24005)	30 (34805)	$3.2nnz$
ISAI4	4 (1115)	8 (5215)	12 (12315)	16 (22415)	20 (35515)	24 (51615)	$4.7nnz$
ISAI5	3 (1435)	7 (7035)	10 (16835)	13 (30835)	17 (49035)	20 (71435)	$6.5nnz$

TABLE 4.3

Iteration count in stationary iteration and density nnz of Block Jacobi (block size 1,2,3,4,5), SAI and ISAI preconditioners (pattern $|A|^k, k = 1, ..., 5$) for the triangular 2D Laplace matrix. The first number denotes the iterations, the second number in brackets the nonzeros of the preconditioner.

matrix	n	nnz(A)	nnz(L)	cond(A)
nos2	957	4137	2547	6.3e9
bcsstk24	3562	159910	81736	1.9e11
bcsstk38	8032	355460	181746	5.5e16
offshore	259789	4242673	2251231	*
af_shell3	504855	17562051	9033453	*
apache2	715176	4817870	2766523	*

TABLE 4.4

Matrices for the numerical test.

count the nonzero entries by blocksize times number of blocks as kN^2. Obviously, the ISAI based iteration is always better than the SAI method with the same density, and also better as the Block Jacobi with comparable density. Allowing denser matrices in ISAI strongly reduces the number of necessary iterations giving improved efficiency on nowadays parallel architectures.

4.3. Triangular Solves: Further Examples. Here, we consider different examples of large ill-conditioned triangular matrices, e.g. constructed by computing ILU factors of symmetric positive definite matrices from the University of Florida Sparse Matrix collection (Table 4.3). All experiments were done in MATLAB. We compare only the (Block) Jacobi and the ISAI preconditioners. In Block Jacobi for $n \times n$ matrix we consider blocksize k, hence kn entries ignoring possible additional sparsity, in ISAI we have patterns $|L|^k$. In all cases the ISAI stationary iteration shows faster convergence as the comparable Block Jacobi method.

5. Conclusions. Based on the close relationship between iterative ILU, DBAI, and SAI methods we have derived different iterative ILU-like algorithms. To solve the resulting sparse triangular systems we have described the Incomplete Sparse Approximate Inverse preconditioner that is closely related to iterative ILU, DBAI and SAI. Furthermore, we have shown that ISAI is especially well suited for triangular linear systems and parallel architectures, and gives fast convergence in stationary iterations.

nos2	JAC	BJAC5	BJAC10	ISAI1	ISAI2	ISAI3
it (nnz)	319 (957)	192 (4779)	96 (9549)	160 (2547)	107 (4132)	80 (5712)

TABLE 4.5

Iteration count in stationary iteration and density of Block Jacobi and ISAI preconditioners for the nos2 example.

bcsstk24	JAC	BJAC5	BJAC10	ISAI1	ISAI2	ISAI3	ISAI4
it (nnz)	* (3562)	162 (17804)	84 (3.5e4)	64 (8.2e4)	40 (1.9e5)	28 (3.4e6)	20 (5.14e6)

TABLE 4.6

Iteration count in stationary iteration and density of Block Jacobi, SAI and ISAI preconditioners for the bcsstk24 example. Here, $*$ denotes no convergence after 1000 iterations.

REFERENCES

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

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

[3] O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press, 1996.

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

[5] 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.

[6] E. CHOW AND A. PATEL, *Fine-grained parallel incomplete LU factorization*, SIAM J. Sci. Comput., 37 (2015), pp. C169–C193.

[7] E. CHOW AND SAAD, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[8] 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.

[9] M. J. GROTE AND T. HUCKLE, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

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

[11] T. HUCKLE AND A. KALLISCHKO, *Frobenius Norm Minimization and Probing for Preconditioning*, International Journal of Computer Mathematics 84(8) (2007), pp. 1225-1248.

[12] L. YU. KOLOTILINA AND A. YU. YEREMIN, *Factorized sparse approximate inverse preconditionings I. Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 4558.

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

[14] 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 (1977), pp. 148162.

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

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

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

[18] T. HUCKLE AND M. SEDLACEK, *Smoothing and regularization with modified sparse approximate inverses*, J. Electr. Comput. Eng., Volume 2010, Article ID 930218, 16 pages.

[19] R. M. HOLLAND, A. J. WATHEN, AND G. J. SHAW, *Sparse approximate inverses and target matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1000-1011.

[20] M. M. WOLF, M. A. HEROUX, AND E. G. BOMAN, *Factors impacting performance of multithreaded sparse triangular solve*, High Performance Computing in Computational Science, VECPAR 2010, 6449 (2010), pp. 32–44.

bcsstk38	JAC	BJAC5	BJAC10	ISAI1	ISAI2	ISAI3	ISAI4
it (nnz)	* (8032)	* (40154)	* (8.0e4)	86 (1.8e5)	46 (5.6e5)	33 (1.2e6)	24 (2.05e6)

TABLE 4.7

Iteration count in stationary iteration and density of Block Jacobi and ISAI preconditioners for the bcsstk38 example. Here, $*$ denotes no convergence after 1000 iterations.

offshore	JAC	BJAC5	BJAC10	ISAI1	ISAI2
it (nnz)	35 (2.6e5)	35 (1.3e6)	33 (2.6e6)	13 (2.25e6)	8 (8.9e6)

TABLE 4.8

Iteration count in stationary iteration and density of Block Jacobi and ISAI preconditioners for the offshore example.

af_shell3	JAC	BJAC5	BJAC10	BJAC20	ISAI1	ISAI2
it (nnz)	50 (5.0e5)	42 (2.5e6)	36 (5.0e6)	33 (1.0e7)	20 (9.0e6)	13 (2.0e7)

TABLE 4.9

Iteration count in stationary iteration and density of Block Jacobi and ISAI preconditioners for the af_shell3 example.

apache2	JAC	BJAC5	BJAC10	BJAC20	ISAI1	ISAI2
it (nnz)	24 (7.15e5)	17 (3.6e6)	16 (7.15e6)	16 (1.4e7)	12 (2.8e6)	8 (6.7e6)

TABLE 4.10

Iteration count in stationary iteration and density of Block Jacobi and ISAI preconditioners for the apache2 example.