Technische Universität Chemnitz

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Matthias Bollhöfer

A Robust ILU Based on Monitoring the Growth of the Inverse Factors

Preprint SFB393/00-31

Abstract

An incomplete LU decomposition with pivoting is presented that progressively monitors the growth of the inverse factors of L, U. The information on the growth of the inverse factors is used as feedback for dropping entries in L and U. This method yields a robust preconditioner in many cases and is often effective especially when the system is highly indefinite. Numerical examples demonstrate the effectiveness of this approach.

Keywords: sparse matrices, ILU, sparse approximate inverse, $AINV$, pivoting, condition estimator.

AMS subject classification: 65F05, 65F10, 65F50.

Preprint-Reihe des Chemnitzer SFB 393

SFB393/00-31 **July 2000**

Contents

Author's addresses:

Matthias Bollhöfer
Fakultät für Mathematik
TU Chemnitz
D–09107 Chemnitz

`http://www.tu-chemnitz.de/~bolle/`

1 Introduction

We consider problems of the form

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

with $A \in \mathbb{R}^{n,n}$ nonsingular and $b \in \mathbb{R}^n$. We focus on problems where A is sparse and where we do not have much information about the system beforehand. These systems might be highly indefinite or ill–conditioned. Since often these systems are very large solving them is a challenge for numerical algorithms. Sometimes it is exceedingly difficult to solve them by iterative techniques and in these cases direct solvers might be preferred. However, there are situations in which 'general purpose' or 'black–box' iterative solvers are required. The most popular and promising iterative techniques so far are preconditioned Krylov–subspace solvers, see, e.g., [15, 22, 12]. Among many techniques, preconditioners based on incomplete LU–factorizations, see e.g., [17, 18, 19] are known to give excellent results for many important classes of problems, such as those arising from the discretization of elliptic partial differential equations.
Nevertheless, there are still many situations where incomplete LU decomposition give poor results. One often has to play around with the parameters, e.g., to adapt a drop tolerance in the incomplete LU decomposition to obtain a successful preconditioner. This is time–consuming since for any problem one has to select the correct values. This reduces the flexibility as a 'black–box' solver. In addition by decreasing parameters to obtain a successful preconditioner we might get enormous fill–in or an unacceptable computational time. In this case direct solvers are the only alternative.
The intention of this paper is to take a closer look at incomplete LU decompositions and especially on how entries are dropped. The main key used here for analyzing dropping in the incomplete LU decomposition is its strong relation [7, 8] to factored sparse approximate inverse methods [3, 4, 2, 16, 21]. In an earlier paper [8] comparisons between an incomplete LU decompositions with pivoting and a factored approximate inverse with pivoting have shown several examples where the approximate inverse was superior to the ILU. So apparently ILUs may gain more stability from approximate inverses by taking a close look at their relations and especially at the way how entries are dropped.

The main idea is to monitor the growth of the inverse factors of L, U while computing L, U and to use this information as feedback for a refined dropping strategy for the entries of L and U.

2 A simple ILU approach

We start with a simple description of a class of incomplete LU factorizations. For the solution of (1) we construct an approximate decomposition

$$A \approx LDU,$$

where $L, U^\top$ are lower triangular matrices with unit diagonal and D is diagonal. One way to construct these decompositions is to partition A as

$$A = \begin{bmatrix} B & F \\ E & C \end{bmatrix} \in \mathbb{R}^{n,n}$$

with $B \in \mathbb{R}$ and the other blocks have corresponding size. Then A is factored as

$$\begin{bmatrix} B & F \\ E & C \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ L_E & I \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} D_B & 0 \\ 0 & S \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 1 & U_F \\ 0 & I \end{bmatrix}}_{U}, \tag{2}$$

where

$$S = C - L_E D_B U_F \in \mathbb{R}^{n-k,n-k} \tag{3}$$

denotes the so–called Schur–complement. The exact LU–decomposition of A (if it exists) can be obtained by successively applying (2) to the Schur–complement S. Even if there exists a decomposition (2) for A and for S, there is no need to compute L_E, U_F, S exactly when constructing a preconditioner. A common approach for reducing fill–in consists of discarding entries in L_E, U_F of small size and defining the approximate Schur–complement only with these sparsified vectors $\tilde{L}_E, \tilde{U}_F$. Here we will concentrate on

$$\tilde{S} = B - \tilde{L}_E F - \left(E - \tilde{L}_E B\right) \tilde{U}_F \tag{4}$$

as one possible definition of an approximate Schur–complement. Equation (4) can be obtained from the lower right block of $\tilde{L}^{-1} A \tilde{U}^{-1}$.
We use the MATLAB notation [1] for convenience. For two integers k, l, $k : l$ denotes the sequence $(k, k+1, \dots, l)$ with the convention that whenever $k > l$ the set is empty. For a matrix $A = (A_{ij})_{i=1,\dots,m,\, j=1,\dots,n}$, we define

$$A_{k:l,q:r} := (A_{ij})_{i=k,\dots,l,\, j=q,\dots,r}.$$

The notation : as a subscript indicates that all columns/rows entries are taken. Thus, $A_{:,2}$ denotes the second column of A and $A_{2,:}$ denotes its second row. Similarly for a nonempty set $s \subseteq \{1, \dots, m\}$ we denote by $A_{s,:}$ the matrix $(A_{ij})_{i \in s,\, j=1,\dots,n}$. With this notation the associated ILU algorithm is roughly as follows.

Algorithm 1 (Incomplete LU factorization (ILU))
Given $A = (A_{ij})_{ij} \in \mathbb{R}^{n,n}$ and a drop tolerance $\tau \in [0,1]$. Compute $A \approx LDU$.

$L = U = I, S = A, D_{11} = S_{11}$.
for $i = 1 : n-1$
  $p_{i+1:n} = S_{i+1:n,i}^\top / S_{ii}$, $q_{i+1:n} = S_{i,i+1:n} / S_{ii}$
  Drop all entries $|p_i|, |q_i|$ if they are less than τ.
  $L_{i+1:n,i} = p_{i+1:n}^\top$, $U_{i,i+1:n} = q_{i+1:n}$.
  $S_{i+1:n,i+1:n} = S_{i+1:n,i+1:n} - L_{i+1:n,i} D_{i,i+1:n} - (S_{i+1:n,i} - L_{i+1:n,i} S_{ii}) U_{i,i+1:n}$
  $D_{i+1,i+1} = S_{i+1,i+1}$
end

Practical versions of incomplete LU decompositions are typically implemented in a slightly different way. It is usually not advisable to update the whole $S_{i+1:n,i+1:n}$ by a rank–one or rank–two modification. Instead, typically only the leading row of $S_{i+1:n,i+1:n}$ is computed, and the transformations on the other rows are post–poned. This corresponds to the so–called I,K,J version of Gaussian elimination[21]. Besides saving memory, this approach is easier to implement since all updates and modifications are performed only once for each row. Thus one can use simple sparse row storage schemes, e.g. the Compressed Sparse Row (CSR) format [21].

Algorithm 2 (Incomplete LU factorization, I, K, J version)
Given $A = (A_{ij})_{ij} \in \mathbb{R}^{n,n}$ *and a drop tolerance* $\tau \in [0, 1]$*. Compute* $A \approx LDU$*.*

$L = U = I, S = A, \mathcal{C} = \mathcal{R} = \emptyset.$
for $i = 1 : n$
    $w = A_{i,:}$
    for $j = 1, \dots, i-1$ *and when* $w_k \neq 0$
        $w_j = w_j / D_{jj}$
        if $|w_j| \leqslant \tau$, $w_j = 0$, ***else*** $w_{j+1:n} := w_{j+1:n} - w_j U_{j,j+1:n}$
    end
    for *all* $j > i$: ***if*** $|w_j / w_i| \leqslant \tau$, $w_j = 0$
    Define $D_{ii} = w_i, \quad U_{i,i:n} = w_{i:n} / D_{ii}, \quad L_{i,1:i-1} = w_{1:i-1}$
end

Mathematically Algorithm 2 can be read as a special version of Algorithm 1, if the approximate Schur–complement is replaced by

$$S_{i+1:n,i+1:n} = S_{i+1:n,i+1:n} - L_{i+1:n,i} D_{i,i} U_{i,i+1:n}.$$

Clearly this replacement would also end up in an exact LU decomposition once we do not drop entries anymore.

3 Stabilized ILU

One problem in dropping entries in Algorithm 1 or Algorithm 2 is that we do not have control of the changes which are affected by dropping. One way to get a more reliable dropping criterion is to take the norm of the i–th row of A into account, e.g. replace τ by $\tau \cdot \|A_{i,:}\|_1$. This is essentially what the ILUT–Algorithm [19] does. A slightly refined version of this strategy, at least if the information on the Schur–complement is available, could be to consider the norm of the i–th row of the Schur–complement as well. This makes sense especially when the corresponding row of the Schur–complement has significantly smaller entries. I.e., instead of dropping entries that are less than τ or $\tau \cdot \|A_{i,:}\|_1$ in absolute value, we could drop entries that are less than $\tau \cdot \min\{\|A_{i,:}\|_1, \|S_{i,i:n}\|_1\}$ in absolute value. Often both choices are a very good compromise but clearly there may still be cases where we could end up in a poor preconditioner.

Algorithm 1, 2 can be supplemented with pivoting. When column pivoting is added to Algorithm 2 it essentially corresponds to the ILUTP–Algorithm which is part of SPARSKIT, see e.g. [21, 20]. So far we have ignored this option to have more clear presentation. Later on, we will return to this point and finally include pivoting. For simplicity let us consider the algorithms without pivoting at this stage.

Recently it has been shown in [7] that Algorithm 1 has a strong relation to sparse approximate inverse preconditioners. Without going into the details, we will roughly describe the idea of AINV–type algorithms [3, 4, 2, 8]. The idea is to directly compute upper triangular matrices W, Z such that $W^\top AZ = D$, with a diagonal matrix D. The version which we will focus on is the so–called right looking AINV, where W and Z are updated by a rank–1 update. Essentially a biorthogonalization process for W and Z is performed, in which $W^\top A$ and $Z^\top A^\top$ are transformed step by step to upper triangular form. Clearly this only holds if no dropping is applied to W, Z.

Algorithm 3 (Factored Approximate INVerse, rank–1 update version)
Given $A = (A_{ij})_{ij} \in \mathbb{R}^{n,n}$ and a drop tolerance $\tau \in [0, 1]$. Compute $A^{-1} \approx ZD^{-1}W^\top$.

$p = q = (0, \ldots, 0) \in \mathbb{R}^n, Z = W = I_n, \mathcal{C} = \mathcal{R} = \emptyset.$
for $i = 1 : n$
  $p_{i:n} = Z_{:,i}^\top A_{i:n,:}^\top,\ q_{i:n} = W_{:,i}^\top A_{:,i:n}$
  Set $p_{i+1:n} := p_{i+1:n}/p_i,\ q_{i+1:n} := q_{i+1:n}/q_i$
  $W_{:,i+1:n} = W_{:,i+1:n} - W_{:,i}p_{i+1:n},\ Z_{:,i+1:n} = Z_{:,i+1:n} - Z_{:,i}q_{i+1:n}$
  Drop entries W_{kl} *of* $W_{1:i,i+1:n}$, *if* $|W_{kl}| \leqslant \tau$
  Drop entries Z_{kl} *of* $Z_{1:i,i+1:n}$, *if* $|Z_{kl}| \leqslant \tau$
end
Choose diagonal entries of D as the components of p (or equivalently of q).

In principle we could modify Algorithm 1 such that the inverses of its triangular factors L, U are computed on the fly. For this purpose we supplement Algorithm 1 with a progressive inversion of L, U. At step $i - 1$, U is of the form

$$U = \begin{bmatrix} U_{1:i-1,1:i-1} & U_{1:i-1,i:n} \\ \mathrm{O} & I \end{bmatrix}$$

and the i-th step will compute the entries $U_{i,i+1:n}$ and add them to the current U to get U_{new}. Let $q^\top$ be the row vector $q^\top = U_{i,:} - e_i^\top$. Note that the 'diagonal' element q_i of q is zero. Then,

$$U_{new} = U + e_i q^\top = (I + e_i q^\top)U.$$

It follows that

$$U_{new}^{-1} = U^{-1}(I + e_i q^\top)^{-1} = U^{-1}(I - e_i q^\top)\ .$$

Of course analogous arguments hold for L. This provides a formula for progressively computing $L^{-\top}, U^{-1}$ throughout the algorithm. We call the inverse factors Z, W as in Algorithm 3. With these additional factors Z, W and a modified Schur–complement it was shown in [7] that the supplemented version of Algorithm 1 is essentially equivalent to Algorithm 3.

Theorem 4 *Suppose that Algorithm 1 is supplemented with a progressive inversion of L, U. Suppose in addition that in step i of Algorithm 1 an entry L_{ji} is discarded only if $|L_{ji}| \max\{1\} \cup \{|W_{ki}| : k < i\} \leqslant \tau$, $i = 1, \dots, n$. Suppose that in Algorithm 1 and 3 W_{kl} is dropped from $W_{1:i,i+1:n}$ if $|W_{kl}| \leqslant \tau$. If the (modified) Schur–complement $S_{i+1:n,i+1:n}$ is defined via*

$$S_{i+1:n,i+1:n} = W_{:,i+1:n}^\top A_{:,i+1:n},$$

then we have for any $k > l$:

$$|(L^{-\top})_{kl} - W_{kl}| \leqslant \tau(2(k-l) - 1)$$

and the diagonal entries of D are those of p.

Proof. See [7]. □

The most interesting point about this relation is that Theorem 4 requires to modify the dropping strategy for L (and similarly for U). Now typically applying dropping to sparse approximate inverse factors is less harmful than for incomplete LU decompositions, because in dropping small entries of size τ in W, Z the effective error in $W^\top AZ$ is only between linear and quadratic with respect to τ. And $W^\top AZ$ is the matrix which needs to be transformed to an approximately diagonal matrix D. On the other hand if we apply dropping to the factors L, U of an ILU the related effect is rational since we do not know in advance the effect for $L^{-1}AU^{-1}$. But for preconditioning, this is precisely what we need to know. So if we can construct an ILU that is somehow almost equivalent to an approximate inverse, then we might hope that dropping is more reliable and the resulting preconditioner is much more efficient for those situations where dropping has a serious impact on the quality of the preconditioner. Numerical results in [8] illustrate that for some extremely indefinite and ill–conditioned problems the approximate inverse behaves better than an ILU.

To turn the result of Theorem 4 into an algorithm we will certainly not invert L, U in Algorithm 1. Let us take a look at the criterion for dropping entries in L. We need to know $\max\{1\} \cup \{|W_{ki}| : k < i\}$, which means we need to know the i–th row of L^{-1}, i.e., $W_{1:i-1,i} = (L^{-1})_{i,1:i-1}$. At least it would be convenient to have an estimate for $\|(L^{-1})_{i,1:i-1}\|_1$ which could serve as a substitute for $\{|W_{ki}| : k < i\}$. To do this we use a general condition estimator for triangular matrices from [14, 9] as a helpful estimate for $\|(L^{-1})_{i,1:i-1}\|_1$. This condition estimator is based on solving a system with an upper triangular matrix U where the right hand side y only consists of ± 1 and the signs are chosen to successively maximize the solution x of $Ux = y$. Another look at this condition estimator shows that the components of $x = U^{-1}y$ precisely estimate $\|(U^{-1})_{i,i:n}\|_1$. To adapt this estimator to our problem we will consider $Lx_L = y_L$ and $U^\top x_U = y_U$ to get estimates for $\|(L^{-1})_{i,1:i-1}\|_1$ and $\|(U^{-1})_{1:i-1,i}\|_1$.

Algorithm 5 (Condition Estimator for (L^{-1}) adapted from [14, 9])
Let $L = (L_{ij})_{ij} \in \mathbb{R}^{n,n}$ be unit lower triangular. Compute $Lx = y$, where $y^\top \in (\pm 1, \cdots, \pm 1)$.

$p = p_+ = p_- = x = (0, \dots, 0)^\top \in \mathbb{R}^n$, *and let $\nu = 0$ be the associated 1–norm of p*

for $i = 1 : n$

$x_+ = 1 - p_i,\ x_- = -1 - p_i$

Let s be the set of nonzero components of $L_{i+1:n,i}$

$p_{+,s} = p_s + L_{s,i}x_+,\ p_{-,s} = p_s + L_{s,i}x_-$

$\nu_+ = \nu - \|p_s\|_1 + \|p_{+,s}\|_1,\ \nu_- = \nu - \|p_s\|_1 + \|p_{-,s}\|_1$

if $|x_+| + \nu_+ > |x_-| + \nu_-$

$x_i = x_+,\ \nu = \nu_+$

$p_s = p_{+,s},\ p_{-,s} = p_{+,s}$

else

$x_i = x_-,\ \nu = \nu_-$

$p_s = p_{-,s},\ p_{+,s} = p_{-,s}$

end

end

In principle one could also use different condition estimators, e.g. [5, 6]. But what we really need is not an estimate for the norm of L^{-1} but an estimate for the norm of each row of L^{-1}. From this point of view to take as right hand side a vector y which only consists of ± 1 is reasonable and is more attractive for this problem.

Now we can supplement Algorithm 1 with the condition estimator Algorithm 5 applied to the L and $U^\top$ factors of the ILU and the components of $x_L = L^{-1}y_L, x_U = U^{-1}y_u$ are serving as estimates for $(L^{-1})_{i,1:i-1}$, $(U^{-1})_{1:i-1,i}$. We still have to discuss the choice of the approximate Schur–complement. Although Theorem 4 is based on defining the approximate Schur–complement via $S_{i+1:n,i+1:n} = W^\top_{:,i+1:n}A_{:,i+1:n}$, it can be seen in the proof of this Theorem that an analogous relation will hold for the case where p, q of Algorithm 3 are defined via

$$p_{i:n} = Z^\top_{:,i}A^\top W_{i:n,:},\ q_{i:n} = W^\top_{:,i}AZ_{:,i:n}. \tag{5}$$

In this case the related Schur–complement for Theorem 4 is

$$S_{i+1:n,i+1:n} = W^\top_{:,i+1:n}AZ_{:,i+1:n}. \tag{6}$$

In fact, when Algorithm 3 is supplemented with pivoting, (5) is used to ensure that $p_i = q_i \neq 0$. Furthermore (6) is related to the choice of S in Algorithm 1 since it consists of taking the lower right block of $\tilde{L}^{-1}A\tilde{U}^{-1}$. Clearly the definition of the approximate Schur–complement in Algorithm 1 is not precisely the same as (6). But according to [7], one has a close connection to Algorithm 3 with this choice of an approximate Schur–complement if dropping is applied in slightly different way.

As a next step to define the ILU we introduce pivoting. We define permutation vectors π, σ, such that $A(\pi, \sigma) = LD(\pi, \sigma)U$ provided that no dropping is applied. In principle, applying permutation matrices Π, Σ to (2), changes this equation to

$$\begin{pmatrix} I & O \\ O & \Pi^\top \end{pmatrix} \begin{bmatrix} B & F \\ E & C \end{bmatrix} \begin{pmatrix} I & O \\ O & \Sigma \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ \Pi^\top L_E & I \end{bmatrix} \begin{bmatrix} D_B & 0 \\ 0 & \Pi^\top S\Sigma \end{bmatrix} \begin{bmatrix} 1 & U_F\Sigma \\ 0 & I \end{bmatrix}.$$

This illustrates how S, L and U have to be adapted. It is clear that if we include the condition estimator, analogous changes are made. It should also be obvious that in practice one will not physically interchange rows of L and columns of U but instead one uses index vectors.

In principle we can introduce a pivoting process to Algorithm 1 which ensures that in the permuted matrix $|p_i| \geqslant \alpha \max_{j=i+1,\dots,n} |p_j|$ and $|q_i| \geqslant \alpha \max_{j=i+1,\dots,n} |q_j|$. This guarantees that after the division by p_i, q_i the entries of p_j/p_i, q_j/q_i are less than $1/\alpha$ in absolute value. Here the parameter $\alpha \in [0,1]$ is chosen a priori. The choice $\alpha = 1$ refers to strict pivoting, i.e. the maximum entry in absolute value will become p_i or q_i, while any smaller choice of α causes only pivoting if the diagonal entry is much smaller than the maximum entry of $|p_{i+1:n}|, |q_{i+1:n}|$. Now we can go one step further and use the freedom in the choice of pivots to add a strategy of Markowitz type [10], i.e., we consider the set of pivots $|p_k|$ that are larger than $\alpha \max_{j=i+1,\dots,n} |p_j|$ and among these we take the one with the minimum fill–in. This is a typical strategy to maintain sparsity in the Schur–complement when using direct methods [10]. To do this, replace $\max_{j=i+1,\dots,n} |p_j|$ by z and define a set $\mathrm{piv}(p)$ by

$$\mathrm{piv}(p,z) = \{k : \ |p_k| \geqslant \alpha z\}. \tag{7}$$

For any k, let $\mathrm{nnzc}_i(k)$ denote the number of nonzeros of $S_{i:n,k}$ and let $\mathrm{nnzr}_i(k)$ denote the number of nonzeros of $S_{k,i:n}$. As pivot we will choose $j \in \mathrm{piv}(p,z)$ such that

$$\mathrm{nnzc}_i(j) = \min_{k \in \mathrm{piv}(p,z)} \mathrm{nnzc}_i(k). \tag{8}$$

I.e., among all admissible pivots choose the one which locally minimizes the fill–in. The same process needs to be repeated for q. In theory this process needs to be alternated between p and q because we have to make sure the diagonal pivots are not getting smaller. For this reason we always increase z. A local pivoting step can then look as follows.

Algorithm 6 (Local pivoting with respect to fill–in)
Given $A = (A_{ij})_{ij} \in \mathbb{R}^{n,n}$ *and a pivoting tolerance* $\alpha \in [0,1]$.
Let $S_{i:n,i:n}$ *denote the Schur–complement on entry to step i of Algorithm 1.*

$z = 0$
while *pivots not satisfactory*
  $p_{i:n} = S_{i,i:n}, \ \ z = \max\{z, \max_{j=i,\dots,n} |p_j|\}$
  Choose $\mu \in \mathrm{piv}(p,z)$ *such that* $\mathrm{nnzc}_i(\mu)$ *is minimal.*
  Interchange columns/components i, μ *of* $p, \sigma, S_{i:n,:}, U_{1:i-1,:}$
  $q_{i:n} = S_{i:n,i}, \ \ z = \max\{z, \max_{j=i,\dots,n} |q_j|\}$
  Choose $\mu \in \mathrm{piv}(q,z)$ *such that* $\mathrm{nnzr}_i(\mu)$ *is minimal.*
  Interchange columns/components i, μ *of* $q, \pi, S_{:,i:n}, L_{:,1:i-1}$
end

The while loop only terminates if no more interchanges are performed.

Together with the condition estimator in Algorithm 5, Algorithm 6 is used to stabilize the incomplete LU decomposition from Algorithm 1. We summarize these changes to a new ILU algorithm.

Algorithm 7 (Stabilized Incomplete LU factorization ($ILUSTAB$))
Given $A = (A_{ij})_{ij} \in \mathbb{R}^{n,n}$, a drop tolerance $\tau \in [0,1]$ and a pivoting tolerance $\alpha \in [0,1]$. Compute $A(\pi,\sigma) \approx LDU$.

$L = U = I, S = A, D_{11} = S_{11}, \pi = \sigma = (1,\dots,n)$.
$x_L = p_L = p_{+,L} = p_{-,L} = x_U = p_U = p_{+,U} = p_{-,U} = (0,\dots,0)^\top \in \mathbb{R}^n$, $\nu_L = \nu_U = 0$.
for $i = 1 : n-1$

Apply Algorithm 6.
Whenever p requires pivoting, interchange $p_U, p_{+,U}, p_{-,U}, x_U$ as well
Whenever q requires pivoting, interchange $p_L, p_{+,L}, p_{-,L}, x_L$ as well

$L_{i+1:n,i} = p_{i+1:n}^\top$, $U_{i,i+1:n} = q_{i+1:n}$.

Apply step i of Algorithm 5 for L with
ν, x, p, p_+, p_- replaced by $\nu_L, x_L, p_L, p_{+,L}, p_{-,L}$
Apply step i of Algorithm 5 for $U^\top$ with
ν, x, p, p_+, p_- replaced by $\nu_U, x_U, p_U, p_{+,U}, p_{-,U}$

drop all entries $|L_{ji}|$ of $L_{i+1:n,i}$, if $|L_{ji}| \max\{1, |x_{L,i}|\} \leqslant \tau \min\{\|A_{i,:}\|_1, \|S_{i,i:n}\|_1\}$
drop all entries $|U_{ji}|$ of $U_{i,i+1:n}$, if $|U_{ij}| \max\{1, |x_{U,i}|\} \leqslant \tau \min\{\|A_{i,:}\|_1, \|S_{i,i:n}\|_1\}$

$S_{i+1:n,i+1:n} = S_{i+1:n,i+1:n} - L_{i+1:n,i}S_{i,i+1:n} - (S_{i+1:n,i} - L_{i+1:n,i}S_{ii})\, U_{i,i+1:n}$
$D_{i+1,i+1} = S_{i+1,i+1}$

end

The two major differences between Algorithm 1 and Algorithm 7 are the application of pivoting and the inclusion of a condition estimator. The latter is motivated by the strong relations between incomplete LU factorizations and factored approximate inverse preconditioners.

4 Numerical Results

This section presents numerical experiments to validate the algorithms. So far, Algorithm 7 is implemented in MATLAB [1].

- The matrices are initially reordered using the symmetric minimum degree ordering [13].
- An a priori scaling is used such any row of the given matrix has unit 1–norm.
- For the pivoting process $\alpha = 0.1$ is used.
- Different values were used for the drop tolerance $\tau = 0.1, 0.3$.

For the numerical experiments several unsymmetric matrices from the Harwell–Boeing Collection [11] were chosen.

The result are compared with

- LU from **MATLAB** also with pivoting tolerance $\alpha = 0.1$
- *LUINC* from **MATLAB** with $\alpha = 0.1$ and drop tolerances $\tau = 0.1$, 0.01, 10^{-3}, 10^{-4}, 10^{-5}
- $ILUTP$ from SPARSKIT using the same tolerance $\alpha = 0.1$ for pivoting but $\tau = 0.1$, 0.01, 10^{-3}, 10^{-4}, 10^{-5} for dropping.

The numerical results for $ILUTP$ [21] were performed on an SGI workstation with two 190 MHz R10000 (IP25) processors under IRIX 6.2 and 512 MB memory.

As iterative solvers GMRES(30) [22] is used. The iteration was stopped after the residual norm was less than $\sqrt{\text{eps}}$ times the initial residual norm, where eps $\approx 2.2204 \cdot 10^{-16}$ denotes the machine precision. The iteration was stopped after 500 steps. Every iterative solution which broke down or did not converge within the number of steps was noted as a failure.

We briefly describe the results for several matrices and then give detailed numerical results for several selected examples.

To give a rough idea on how the method performed on the Harwell–Boeing collection we simply summarize in Table 1 which method successfully solved how many problems with respect to the drop tolerance τ. The tests were done on 94 matrices from the Harwell-Boeing collection.

Table 1: **Summary of results — Successful Computation**

Harwell–Boeing Collection (94 test matrices)						
Preconditioner	Drop tolerance τ					
	0.3	0.1	0.01	10^{-3}	10^{-4}	10^{-5}
ILUSTAB	89	92				
LUINC		31	52	68	79	87
ILUTP		53	69	78	84	90

Note that there were only two matrices which could not be solved with $ILUSTAB$ for $\tau \in \{0.3, 0.1\}$. These are the matrices *facsimile/fs7603*, *grenoble/gre216b*. These matrices could be solved with $\tau = 0.01$. But LUINC could also not solve *facsimile/fs7603* and for *facsimile/fs7603* ILUTP needed $\tau = 10^{-4}$. For *grenoble/gre216b* LUINC and ILUTP needed $\tau = 10^{-5}$.

We now comment on several matrices from the Harwell–Boeing–Collection. This collection consists of many matrices from different areas. Related matrices are put together in a group and comments are done with respect to these groups. For some selected examples we will show separate tables. In each table (e.g., Table 2) we will present the the choice of the drop tolerance τ and the related fill–in factor (that is the ratio of the number of nonzeros of $L + U$ divided by the number of nonzeros of A). Next the number of iteration steps using GMRES(30) is shown. For the **MATLAB** algorithms LU, ILUSTAB and LUINC we use the flop count as measure for the number of operations. The flop count is split into the flops required for the decomposition and the flops to solve a linear system using GMRES(30).

- CHEMWEST: These matrices are some of those for which LUINC and ILUTP needed smallest drop tolerances to be successful while *ILUSTAB* was able to solve all of them already for $\tau = 0.3$. Detailed results for the three biggest WEST–matrices are given in Table 2, 3, 4.

Table 2: **Matrix CHEMWEST/WEST0989**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		3.2	1	$1.2 \cdot 10^5$	$9.9 \cdot 10^4$
ILUSTAB	0.3	1.3	20	$1.8 \cdot 10^5$	$1.4 \cdot 10^6$
	0.1	1.5	14	$1.7 \cdot 10^5$	$8.3 \cdot 10^5$
LUINC	10^{-1}	0.7	—	$1.1 \cdot 10^4$	—
	10^{-2}	1.0	—	$1.4 \cdot 10^4$	—
	10^{-3}	1.2	—	$2.0 \cdot 10^4$	—
	10^{-4}	1.6	—	$3.8 \cdot 10^4$	—
	10^{-5}	1.9	6	$4.7 \cdot 10^4$	$2.7 \cdot 10^5$
ILUTP	10^{-1}	1.0	—		
	10^{-2}	1.4	—		
	10^{-3}	1.9	—		
	10^{-4}	2.4	309		
	10^{-5}	2.7	10		

- FACSIMILE: LUINC from **MATLAB** could not solve most of these matrices for $\tau = 0.1, 0.01$. For $\tau = 10^{-3}$ it was able to solve 50% of them and for $\tau = 10^{-4}, 10^{-5}$ only *fs1836*, *fs7602*, *fs7603* could not be solved. For those problems that could be solved, the fill–in was moderate and the number of iteration steps was small.
  In contrast to this ILUSTAB could solve all of these matrices already for $\tau = 0.3$ except *fs7603* which could not be solved. The fill–in was small as well. The number of iteration steps was small except for *fs7602* which required 60 steps for $\tau = 0.3$ and 31 for $\tau = 0.1$.
  ILUTP solved most of these problems for $\tau = 0.1$. All problems including *fs7603* were solved for $\tau = 10^{-4}, 10^{-5}$.

Table 3: **Matrix CHEMWEST/WEST1505**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		4.2	1	$4.0 \cdot 10^5$	$1.7 \cdot 10^5$
ILUSTAB	0.3	1.4	22	$3.5 \cdot 10^5$	$2.5 \cdot 10^6$
	0.1	1.7	17	$4.0 \cdot 10^5$	$1.7 \cdot 10^6$
LUINC	10^{-1}	0.7	—	$1.7 \cdot 10^4$	—
	10^{-2}	1.0	—	$2.2 \cdot 10^4$	—
	10^{-3}	1.2	—	$3.2 \cdot 10^4$	—
	10^{-4}	1.7	16	$6.8 \cdot 10^4$	$1.5 \cdot 10^6$
	10^{-5}	2.0	6	$8.6 \cdot 10^4$	$4.1 \cdot 10^5$
ILUTP	10^{-1}	1.0	—		
	10^{-2}	1.4	—		
	10^{-3}	1.9	—		
	10^{-4}	2.4	—		
	10^{-5}	2.7	—		

Table 4: **Matrix CHEMWEST/WEST2021**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		5.6	1	$1.2 \cdot 10^6$	$2.7 \cdot 10^5$
ILUSTAB	0.3	1.6	20	$6.8 \cdot 10^5$	$2.9 \cdot 10^6$
	0.1	1.7	14	$6.7 \cdot 10^5$	$1.7 \cdot 10^6$
LUINC	10^{-1}	0.7	—	$2.2 \cdot 10^4$	—
	10^{-2}	0.9	—	$3.0 \cdot 10^4$	—
	10^{-3}	1.2	—	$4.6 \cdot 10^4$	—
	10^{-4}	1.6	—	$8.7 \cdot 10^4$	—
	10^{-5}	1.9	6	$1.2 \cdot 10^5$	$5.5 \cdot 10^5$
ILUTP	10^{-1}	1.0	—		
	10^{-2}	1.4	—		
	10^{-3}	1.9	—		
	10^{-4}	2.5	—		
	10^{-5}	3.1	14		

For those problems that could be solved the fill–in was small. The largest number of iterations were 155 for *fs7602* and $\tau = 0.1$, 62 for *fs7602* and $\tau = 10^{-3}$. For all other methods it was less, if they could be solved at all.

- GEMAT: ILUSTAB could not solve these matrices for $\tau = 0.3$ but for $\tau = 0.1$.
  LUINC could solve these matrices for $\tau = 10^{-4}$ but with roughly four times of the fill–in of ILUSTAB.
  ILUTP could solve these matrices for $\tau = 10^{-3}$ but with more than twice as much fill–in as ILUSTAB. For these matrices the LU decomposition needed more than 70 times of fill than the initial matrix.
  For *gemat12* see Table 5. The results for *gemat11* are similar.

Table 5: **Matrix GEMAT/GEMAT12**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		73.5	1	$1.7{\cdot}10^9$	$1.0{\cdot}10^7$
ILUSTAB	0.3	1.0	—	$1.4{\cdot}10^6$	—
	0.1	1.3	67	$2.0{\cdot}10^6$	$3.5{\cdot}10^7$
LUINC	10^{-1}	0.6	—	$1.8{\cdot}10^5$	—
	10^{-2}	1.4	—	$1.6{\cdot}10^5$	—
	10^{-3}	2.7	—	$1.3{\cdot}10^7$	—
	10^{-4}	5.2	10	$5.4{\cdot}10^7$	$5.7{\cdot}10^6$
	10^{-5}	9.2	5	$1.3{\cdot}10^8$	$3.9{\cdot}10^6$
ILUTP	10^{-1}	1.0	—		
	10^{-2}	2.0	—		
	10^{-3}	3.4	17		
	10^{-4}	5.2	7		
	10^{-5}	7.4	4		

- GRENOBLE: for $\tau = 0.1$ LUINC could only solve *gre115*, *gre216a*, *gre343*, *gre512*. But even for some of those the fill–in factor was already enormous (e.g. 5.9 for *gre216a*, 7.7 for *gre343*, 11.3 for *gre512*). The same problem occurred for the other matrices that could only be solved for smaller τ. All matrices could finally be solved with $\tau = 10^{-5}$.
  ILUSTAB solved all matrices except *gre216b*, *gre1107* for $\tau = 0.3$. The fill–in was slightly better (e.g. i.e. 3.8 for *gre216a*, 4.9 for *gre343*, 7.8 for *gre512*). *gre1107* could be solved with $\tau = 0.1$ but with a fill–in factor 7.4. This was still better than LUINC, which needed $\tau = 10^{-3}$ and produced a fill–in factor 23.0!
  For $\tau = 0.1$, ILUTP could solve *gre115*, *gre185*, *gre216a*. But even then the fill–in factor was sometimes large (i.e. 7.0 for *gre216a*, 12.6 for *gre512*). The same problem

occurred for the other matrices that could only be solved for smaller τ. For example *gre1107* could be solved with $\tau = 10^{-3}$ and a fill–in factor 21.3. All matrices could finally be solved with $\tau = 10^{-5}$.
The problem with the fill–in also extremely affects the sparse *LU* decomposition. For example *gre1107* required a fill–in factor 44.1!
For those problems that could be solved by one of these methods the number of iteration steps was moderate.

- LNS: ILUSTAB solved them all for $\tau = 0.3$. The fill–in was moderate (3.6 for *lns3937* was already maximum) an so was the number of iteration steps (at most 29).
  LUINC could not solve any of these matrices for $\tau = 0.1, 0.01$ but *lns511*, *lnsp511*, *lns3937*, *lnsp3937* for $\tau = 10^{-3}$. The biggest matrices required twice as much fill–in as ILUSTAB.
  ILUTP could solve the two smallest matrices for $\tau = 0.1$ and the medium size matrices for $\tau = 10^{-3}$.
  The two biggest matrices could only be solved for $\tau = 10^{-5}$. For *lns3937* see Table 6. The results for *lnsp3937* were quite similar.

Table 6: **Matrix LNS/LNS3937**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		46.1	1	$2.9 \cdot 10^8$	$4.9 \cdot 10^6$
ILUSTAB	0.3	3.6	28	$2.3 \cdot 10^7$	$1.4 \cdot 10^7$
	0.1	4.9	16	$4.1 \cdot 10^7$	$7.7 \cdot 10^6$
LUINC	10^{-1}	1.0	—	$6.4 \cdot 10^5$	—
	10^{-2}	3.7	—	$1.0 \cdot 10^7$	—
	10^{-3}	7.4	29	$3.2 \cdot 10^7$	$2.0 \cdot 10^7$
	10^{-4}	12.3	9	$6.6 \cdot 10^7$	$7.1 \cdot 10^6$
	10^{-5}	17.0	9	$1.0 \cdot 10^8$	$5.0 \cdot 10^6$
ILUTP	10^{-1}	0.8	—		
	10^{-2}	1.4	—		
	10^{-3}	2.5	—		
	10^{-4}	3.5	—		
	10^{-5}	4.4	—		

- NUCL: ILUSTAB could solve all matrices for $\tau = 0.3$ but the fill–in was poor, e.g., 28.6 for *nnc1374*. The number of iteration steps was at most 28.
  LUINC did not solve any of these matrices for $\tau = 10^{-1}, \ldots, 10^{-5}$.
  ILUTP could solve all the problem for $\tau = 10^{-3}$ and a better fill–in factor than ILUSTAB (e.g. 6.6 for *nnc1374* but 463 iteration steps).

Here the direct solver produced significantly less fill–in for *nnc1374* (factor 14.6) than ILUSTAB.

- PORES: PORES1, PORES3 could be solved by ILUSTAB for $\tau = 0.3$ and ILUTP for $\tau = 0.1$. LUINC needed $\tau = 0.01$ for PORES3. The number of iteration steps was small except for PORES3, $\tau = 0.1$ and ILUTP which needed 248 steps, but for $\tau = 0.01$ the number of steps were small while the fill–in was still below the fill–in of the original matrix. For matrix PORES2 see Table 7.

Table 7: **Matrix PORES/PORES2**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		5.1	1	$2.5{\cdot}10^6$	$2.9{\cdot}10^5$
ILUSTAB	0.3	1.0	54	$6.8{\cdot}10^5$	$6.9{\cdot}10^6$
	0.1	1.2	15	$9.6{\cdot}10^5$	$1.5{\cdot}10^6$
LUINC	10^{-1}	0.5	—	$4.4{\cdot}10^4$	—
	10^{-2}	0.6	113	$6.0{\cdot}10^4$	$1.4{\cdot}10^7$
	10^{-3}	1.1	26	$1.6{\cdot}10^5$	$3.1{\cdot}10^6$
	10^{-4}	1.8	9	$4.1{\cdot}10^5$	$8.3{\cdot}10^5$
	10^{-5}	2.4	5	$6.7{\cdot}10^5$	$4.7{\cdot}10^5$
ILUTP	10^{-1}	0.4	—		
	10^{-2}	0.8	—		
	10^{-3}	1.7	30		
	10^{-4}	3.2	13		
	10^{-5}	4.6	9		

- SAYLOR: SAYLR1/SAYLR3 were solved by ILUSTAB for $\tau = 0.3$ and ILUTP for $\tau = 0.1$. For SAYLR3, LUINC failed for all τ. For SAYLR4 see Table 8.

- SHERMAN: ILUSTAB solved all the matrices for $\tau = 0.3$, but for *sherman3* it needed 138 iteration steps. For the other matrices the iteration count was less than half as much. The fill–in was less than twice as much as the initial fill. The number of iterations was much lower for $\tau = 0.1$ but with more fill–in. LUINC could only solve *sherman4*, *sherman5* for $\tau = 0.1$ and it needed 123 iteration steps for *sherman5*. For $\tau = 0.01$ it needed only a moderate number of iteration steps, but *sherman1* still could not be solved for $\tau = 10^{-2}$. ILUTP could solve all matrices but for *sherman2* it needed $\tau = 10^{-5}$ (see Table 9). For *sherman4* and $\tau = 0.1$ the number of iteration steps (449) was still big. This changed when using $\tau = 0.01$.

Table 8: **Matrix SAYLOR/SAYLR4**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		14.7	1	$3.7{\cdot}10^7$	$1.5{\cdot}10^6$
ILUSTAB	0.3	2.6	44	$1.6{\cdot}10^7$	$1.8{\cdot}10^7$
	0.1	3.1	15	$2.0{\cdot}10^7$	$5.2{\cdot}10^6$
LUINC	10^{-1}	0.6	—	$1.1{\cdot}10^5$	—
	10^{-2}	0.6	—	$1.1{\cdot}10^5$	—
	10^{-3}	0.6	—	$1.1{\cdot}10^5$	—
	10^{-4}	1.6	33	$9.5{\cdot}10^5$	$1.2{\cdot}10^7$
	10^{-5}	2.7	11	$3.5{\cdot}10^6$	$3.1{\cdot}10^6$
ILUTP	10^{-1}	0.6	352		
	10^{-2}	0.6	155		
	10^{-3}	0.6	153		
	10^{-4}	2.4	18		
	10^{-5}	3.5	8		

Table 9: **Matrix SHERMAN/SHERMAN2**

Method / τ		fill–in factor	# it. steps	flops dec.	flops solve
sparse LU		14.0	1	$8.1{\cdot}10^7$	$1.4{\cdot}10^6$
ILUSTAB	0.3	0.4	30	$1.6{\cdot}10^6$	$4.4{\cdot}10^6$
	0.1	0.6	14	$2.6{\cdot}10^6$	$1.7{\cdot}10^6$
LUINC	10^{-1}	0.2	—	$9.7{\cdot}10^4$	—
	10^{-2}	0.4	21	$2.6{\cdot}10^5$	$2.6{\cdot}10^6$
	10^{-3}	0.7	7	$7.7{\cdot}10^5$	$7.5{\cdot}10^5$
	10^{-4}	1.1	5	$2.4{\cdot}10^6$	$6.2{\cdot}10^5$
	10^{-5}	1.7	4	$5.1{\cdot}10^6$	$5.9{\cdot}10^5$
ILUTP	10^{-1}	0.3	—		
	10^{-2}	0.6	—		
	10^{-3}	1.0	—		
	10^{-4}	1.6	—		
	10^{-5}	2.1	61		

The numerical examples have illustrated the robustness of taking the growth of the inverse triangular factors into account when computing an incomplete LU decomposition. Of course ILUSTAB is neither always the most efficient nor always the fastest (with respect to the flops) nor always the ILU with the smallest amount of fill–in. But in many cases it is a pretty good compromise between standard incomplete LU decompositions and the full sparse LU decomposition. In many examples it is not necessary to use a trial–and–error strategy for choosing the drop tolerance. The drop tolerance is automatically adapted with respect to the growth of the inverse factors. In several cases where a direct solver is superior to iterative method (cf.Table 2), 3, 4 with respect to the number of flops, the fill–in for ILUSTAB is still moderate and often even less less than that for LUINC, ILUTP. Conversely on some problems which cause trouble to direct solvers (cf. Table 5) ILUSTAB gains from its sparsity and being used as iterative solver.

The drawback of this algorithm is of course that it is more comparable with sparse direct solvers because it requires explicit knowledge of the Schur–complement. Clearly there are several problems where standard incomplete LU decompositions used as preconditioners give powerful iterative solvers. In these cases apparently ILUSTAB will be slower because one has a certain time consuming overhead for computing and administrating the approximate Schur–complement.

5 Conclusions

A version of an incomplete LU decomposition has been presented that performs dropping with respect to the growth of the inverses of the triangular factors. We have illustrated that the resulting preconditioner is very robust. Often one can avoid adapting the parameters to a specific matrix and still get a preconditioner that is computed in a sensible time with moderate fill–in. For many examples this has turned out to be a good compromise between sparse direct solvers and standard incomplete LU decompositions. Since this preconditioner shares several properties with sparse direct solvers, an implementation based on modified direct solvers seems to be reasonable. Currently codes from direct solvers like the Harwell–Subroutine–Library are under investigation to build this kind of preconditioner. Real–time results for bigger problems will be presented in a forthcoming paper.

References

[1] MATLAB – The language of technical computing. The MathWorks Inc., 1996.

[2] M. Benzi, J. K. Cullum, and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. Technical report LA–UR–99–2899, Los Alamos National Laboratory, Scientific Computing Group, 1999.

[3] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17:1135–1149, 1996.

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

[5] C. H. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl.*, 11(2):312–322, 1990.

[6] C. H. Bischof, J. G. Lewis, and D. J. Pierce. Incremental condition estimation for sparse matrices. *SIAM J. Matrix Anal. Appl.*, 11(4):644–659, 1990.

[7] M. Bollhoefer and Y. Saad. *ILU*s and factorized approximate inverses are strongly related. Part I: Overview of results. Technical Report umsi–2000-39, Minnesota Supercomputer Institue, University of Minnesota, 2000.

[8] M. Bollhoefer and Y. Saad. *ILU*s and factorized approximate inverses are strongly related. Part II: Applications to stabilization. Technical Report umsi–2000-70, University of Minnesota at Minneapolis, Dep. of Computer Science and Engineering, 2000.

[9] A. Cline, C. B. Moler, G. Stewart, and J. Wilkinson. An estimate for the condition number of a matrix. *SIAM J. Numer. Anal.*, 16:368–375, 1979.

[10] I. Duff, A. Erisman, and J. Reid. *Direct Methods for Sparse Matrices.* Oxford University Press, 1986.

[11] I. Duff, R. Grimes, and J. Lewis. Sparse matrix test problems. *ACM Trans. Math. Software*, 15:1–14, 1989.

[12] R. Freund, G. Golub, and N. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, pages 1–44, 1992.

[13] J. A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

[14] G. Golub and C. V. Loan. *Matrix Computations.* The Johns Hopkins University Press, third edition, 1996.

[15] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–436, 1952.

[16] S. Kharchenko, L. Kolotilina, A. Nikishin, and A. Yeremin. A reliable AINV–type preconditioning method for constructing sparse approximate inverse preconditioners in factored form. Technical report, Russian Academy of Sciences, Moscow, 1999.

[17] J. Meijerink and H. A. V. der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m–matrix. *Math. Comp.*, 31:148–162, 1977.

[18] N. Munksgaard. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradient method. *ACM Trans. Math. Software*, 6:206–219, 1980.

[19] Y. Saad. ILUT: a dual treshold incomplete ILU factorization. *Numer. Lin. Alg. w. Appl.*, 1:387–402, 1994.

[20] Y. Saad. SPARSKIT and sparse examples. NA Digest, 1994.

[21] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, 1996.

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

Other titles in the SFB393 series:

99-01 P. Kunkel, V. Mehrmann, W. Rath. Analysis and numerical solution of control problems in descriptor form. January 1999.

99-02 A. Meyer. Hierarchical preconditioners for higher order elements and applications in computational mechanics. January 1999.

99-03 T. Apel. Anisotropic finite elements: local estimates and applications (Habilitationsschrift). January 1999.

99-04 C. Villagonzalo, R. A. Römer, M. Schreiber. Thermoelectric transport properties in disordered systems near the Anderson transition. February 1999.

99-05 D. Michael. Notizen zu einer geometrisch motivierten Plastizitätstheorie. Februar 1999.

99-06 T. Apel, U. Reichel. SPC-PM Po 3D V 3.3, User's Manual. February 1999.

99-07 F. Tröltzsch, A. Unger. Fast solution of optimal control problems in the selective cooling of steel. March 1999.

99-08 W. Rehm, T. Ungerer (Eds.). Ausgewählte Beiträge zum 2. Workshop Cluster-Computing 25./26. März 1999, Universität Karlsruhe. März 1999.

99-09 M. Arav, D. Hershkowitz, V. Mehrmann, H. Schneider. The recursive inverse eigenvalue problem. March 1999.

99-10 T. Apel, S. Nicaise, J. Schöberl. Crouzeix-Raviart type finite elements on anisotropic meshes. May 1999.

99-11 M. Jung. Einige Klassen iterativer Auflösungsverfahren (Habilitationsschrift). Mai 1999.

99-12 V. Mehrmann, H. Xu. Numerical methods in control, from pole assignment via linear quadratic to H_∞ control. June 1999.

99-13 K. Bernert, A. Eppler. Two-stage testing of advanced dynamic subgrid-scale models for Large-Eddy Simulation on parallel computers. June 1999.

99-14 R. A. Römer, M. E. Raikh. The Aharonov-Bohm effect for an exciton. June 1999.

99-15 P. Benner, R. Byers, V. Mehrmann, H. Xu. Numerical computation of deflating subspaces of embedded Hamiltonian pencils. June 1999.

99-16 S. V. Nepomnyaschikh. Domain decomposition for isotropic and anisotropic elliptic problems. July 1999.

99-17 T. Stykel. On a criterion for asymptotic stability of differential-algebraic equations. August 1999.

99-18 U. Grimm, R. A. Römer, M. Schreiber, J. X. Zhong. Universal level-spacing statistics in quasiperiodic tight-binding models. August 1999.

99-19 R. A. Römer, M. Leadbeater, M. Schreiber. Numerical results for two interacting particles in a random environment. August 1999.

99-20 C. Villagonzalo, R. A. Römer, M. Schreiber. Transport Properties near the Anderson Transition. August 1999.

99-21 P. Cain, R. A. Römer, M. Schreiber. Phase diagram of the three-dimensional Anderson model of localization with random hopping. August 1999.

99-22 M. Bollhöfer, V. Mehrmann. A new approach to algebraic multilevel methods based on sparse approximate inverses. August 1999.

99-23 D. S. Watkins. Infinite eigenvalues and the QZ algorithm. September 1999.

99-24 V. Uski, R. A. Römer, B. Mehlig, M. Schreiber. Incipient localization in the Anderson model. August 1999.

99-25 A. Meyer. Projected PCGM for handling hanging in adaptive finite element procedures. September 1999.

99-26 F. Milde, R. A. Römer, M. Schreiber. Energy-level statistics at the metal-insulator transition in anisotropic system. September 1999.

99-27 F. Milde, R. A. Römer, M. Schreiber, V. Uski. Critical properties of the metal-insulator transition in anisotropic systems. October 1999.

99-28 M. Theß. Parallel multilevel preconditioners for thin shell problems. November 1999.

99-29 P. Biswas, P. Cain, R. A. Römer, M. Schreiber. Off-diagonal disorder in the Anderson model of localization. November 1999.

99-30 C. Mehl. Anti-triangular and anti-m-Hessenberg forms for Hermitian matrices and pencils. November 1999.

99-31 A. Barinka, T. Barsch, S. Dahlke, M. Konik. Some remarks for quadrature formulas for refinable functions and wavelets. November 1999.

99-32 H. Harbrecht, C. Perez, R. Schneider. Biorthogonal wavelet approximation for the coupling of FEM-BEM. November 1999.

99-33 C. Perez, R. Schneider. Wavelet Galerkin methods for boundary integral equations and the coupling with finite element methods. November 1999.

99-34 W. Dahmen, A. Kunoth, R. Schneider. Wavelet least squares methods for boundary value problems. November 1999.

99-35 S. I. Solovev. Convergence of the modified subspace iteration method for nonlinear eigenvalue problems. November 1999.

99-36 B. Heinrich, B. Nkemzi. The Fourier-finite-element method for the Lamé equations in axisymmetric domains. December 1999.

99-37 T. Apel, F. Milde, U. Reichel. SPC-PM Po 3D v 4.0 - Programmers Manual II. December 1999.

99-38 B. Nkemzi. Singularities in elasticity and their treatment with Fourier series. December 1999.

99-39 T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: The symmetric case. December 1999.

99-40 T. Penzl. Algorithms for model reduction of large dynamical systems. December 1999.

00-01 G. Kunert. Anisotropic mesh construction and error estimation in the finite element method. January 2000.

00-02 V. Mehrmann, D. Watkins. Structure-preserving methods for computing eigenpairs of large sparse skew-Hamiltonian/Hamiltonian pencils. January 2000.

00-03 X. W. Guan, U. Grimm, R. A. Römer, M. Schreiber. Integrable impurities for an open fermion chain. January 2000.

00-04 R. A. Römer, M. Schreiber, T. Vojta. Disorder and two-particle interaction in low-dimensional quantum systems. January 2000.

00-05 P. Benner, R. Byers, V. Mehrmann, H. Xu. A unified deflating subspace approach for classes of polynomial and rational matrix equations. January 2000.

00-06 M. Jung, S. Nicaise, J. Tabka. Some multilevel methods on graded meshes. February 2000.

00-07 H. Harbrecht, F. Paiva, C. Perez, R. Schneider. Multiscale Preconditioning for the Coupling of FEM-BEM. February 2000.

00-08 P. Kunkel, V. Mehrmann. Analysis of over- and underdetermined nonlinear differential-algebraic systems with application to nonlinear control problems. February 2000.

00-09 U.-J. Görke, A. Bucher, R. Kreißig, D. Michael. Ein Beitrag zur Lösung von Anfangs-Randwert-Problemen einschließlich der Materialmodellierung bei finiten elastisch-plastischen Verzerrungen mit Hilfe der FEM. März 2000.

00-10 M. J. Martins, X.-W. Guan. Integrability of the D_n^2 vertex models with open boundary. March 2000.

00-11 T. Apel, S. Nicaise, J. Schöberl. A non-conforming finite element method with anisotropic mesh grading for the Stokes problem in domains with edges. March 2000.

00-12 B. Lins, P. Meade, C. Mehl, L. Rodman. Normal Matrices and Polar Decompositions in Indefinite Inner Products. March 2000.

00-13 C. Bourgeois. Two boundary element methods for the clamped plate. March 2000.

00-14 C. Bourgeois, R. Schneider. Biorthogonal wavelets for the direct integral formulation of the heat equation. March 2000.

00-15 A. Rathsfeld, R. Schneider. On a quadrature algorithm for the piecewise linear collocation applied to boundary integral equations. March 2000.

00-16 S. Meinel. Untersuchungen zu Druckiterationsverfahren für dichteveränderliche Strömungen mit niedriger Machzahl. März 2000.

00-17 M. Konstantinov, V. Mehrmann, P. Petkov. On Fractional Exponents in Perturbed Matrix Spectra of Defective Matrices. April 2000.

00-18 J. Xue. On the blockwise perturbation of nearly uncoupled Markov chains. April 2000.

00-19 N. Arada, J.-P. Raymond, F. Tröltzsch. On an Augmented Lagrangian SQP Method for a Class of Optimal Control Problems in Banach Spaces. April 2000.

00-20 H. Harbrecht, R. Schneider. Wavelet Galerkin Schemes for 2D-BEM. April 2000.

00-21 V. Uski, B. Mehlig, R. A. Römer, M. Schreiber. An exact-diagonalization study of rare events in disordered conductors. April 2000.

00-22 V. Uski, B. Mehlig, R. A. Römer, M. Schreiber. Numerical study of eigenvector statistics for random banded matrices. May 2000.

00-23 R. A. Römer, M. Raikh. Aharonov-Bohm oscillations in the exciton luminescence from a semiconductor nanoring. May 2000.

00-24 R. A. Römer, P. Ziesche. Hellmann-Feynman theorem and fluctuation-correlation analysis of i the Calogero-Sutherland model. May 2000.

00-25 S. Beuchler. A preconditioner for solving the inner problem of the p-version of the FEM. May 2000.

00-26 C. Villagonzalo, R.A. Römer, M. Schreiber, A. MacKinnon. Behavior of the thermopower in amorphous materials at the metal-insulator transition. June 2000.

00-27 C. Mehl, V. Mehrmann, H. Xu. Canonical forms for doubly structured matrices and pencils. June 2000. S. I. Solov'ev. Preconditioned gradient iterative methods for nonlinear eigenvalue problems. June 2000.

00-29 A. Eilmes, R. A. Römer, M. Schreiber. Exponents of the localization lengths in the bipartite Anderson model with off-diagonal disorder. June 2000.

00-30 T. Grund, A. Rösch. Optimal control of a linear elliptic equation with a supremum-norm functional. July 2000.

The complete list of current and former preprints is available via
`http://www.tu-chemnitz.de/sfb393/preprints.html`.