Multilevel ILU Decomposition

Randolph E. Bank [⋆,1] *and Christian Wagner* [⋆⋆,1,2]

[1] University of California at San Diego, Department of Mathematics, Mail Code 0112, 9500 Gilman Drive, La Jolla, CA 92093-0112, USA.
[2] Institut für Computeranwendungen, Universität Stuttgart, Pfaffenwaldring 27, 70569 Stuttgart, Germany.

Abstract. In this paper, the multilevel ILU (MLILU) decomposition is introduced. During an incomplete Gaussian elimination process new matrix entries are generated such that a special ordering strategy yields distinct levels. On these levels, some smoothing steps are computed. The MLILU decomposition exists and the corresponding iterative scheme converges for all symmetric and positive definite matrices. Convergence rates independent of the number of unknowns are shown numerically for several examples. Many numerical experiments including unsymmetric and anisotropic problems, problems with jumping coefficients as well as realistic problems are presented. They indicate a very robust convergence behavior of the MLILU method.

Key words. Algebraic multi-grid, filter condition, ILU decomposition, iterative method, partial differential equation, robustness, test vector.

AMS subject classifications. 65F10, 65N55.

1 Introduction

In this paper, we consider iterative algorithms

$$u^{(i+1)} = u^{(i)} + M^{-1}(f - K\,u^{(i)}) \tag{1.1}$$

for the solution of linear systems of equations

$$K\,u = f.$$

Although standard multi-grid methods converge independently of the number of unknowns, they show a lack of robustness for certain important problems (see e.g. [11]). To overcome this problem, several multi-grid methods with matrix dependent transfer operators were developed (e.g. [1], [2], [3], [9], [15], [16], [17], [22], [27]). These methods still require a given grid hierarchy. The first truly algebraic multi-grid method (AMG) was introduced by Ruge and Stüben [19]. However, most of these methods are only robust with respect to some difficulties and they are generally based on a similar classical (matrix weighted) interpolation idea.

Recently, Bank and Xu [5] and Bank and Smith [6] (see also [4] and [20]) introduced the hierarchical basis ILU (HBILU) decomposition. The HBILU is actually a modified

⋆ The work of this author was supported by the U. S. Office of Naval Research under contract N00014-89J-1440.
⋆⋆ The work of this author was supported by a fellowship of the NATO scientific council through the DAAD.

ILU decomposition in the spirit of Gustafsson [10] and Wittum [26] which allows some additional fill-in edges. The additional edges produce coarse graphs which are not consistent finite element triangulations, but play a role similar to the coarse grids in a multi-grid method.

The multilevel ILU (MLILU) decomposition may be interpreted as a multi-grid version of the HBILU decomposition. Nevertheless, the main approximation step of the MLILU and the HBILU decomposition are different. Like the HBILU decomposition, the MLILU decomposition uses only information from the system matrix and does not require a given grid hierarchy. The MLILU decomposition represents an incomplete Gaussian elimination, where the fill-in is approximated by several additional matrix entries such that a filter condition is fulfilled. In particular, the interpolation argument which is typical for multi-grid methods is not applied.

In Sect. 2, the MLILU decomposition is defined, followed by a short theoretical investigation in Sect. 3. Sect. 4 introduces the ordering algorithm. Finally, in Sect. 5, we present some numerical experiments.

2 The Decomposition

The MLILU decomposition is defined by a successive elimination of the columns of the system matrix K. In each step, a special elimination technique is applied which limits the fill-in resulting from standard Gaussian elimination. This technique is introduced in Sect. 2.1. After that, a recursive partition of the unknowns yields the multilevel structure of the MLILU decomposition.

2.1 The Approximation Scheme

In this section, we describe the elimination of the first column of the matrix $K^{(i)} \in \mathbb{R}^{n^{(i)} \times n^{(i)}}$

$$K^{(i)} = \begin{pmatrix} d_i & r_i \\ l_i & \bar{K}^{(i)} \end{pmatrix}, \tag{2.1}$$

$d_i \in \mathbb{R}$, $d_i \neq 0$, $r_i \in \mathbb{R}^{1 \times (n^{(i)}-1)}$, $l_i \in \mathbb{R}^{(n^{(i)}-1) \times 1}$, $\bar{K}^{(i)} \in \mathbb{R}^{(n^{(i)}-1) \times (n^{(i)}-1)}$.

Exact Gaussian elimination leads to the factorization

$$K^{(i)} = \begin{pmatrix} 1 & 0 \\ l_i d_i^{-1} & I_{n^{(i)}-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & K_{\mathrm{S}}^{(i)} \end{pmatrix} \begin{pmatrix} d_i & r_i \\ 0 & I_{n^{(i)}-1} \end{pmatrix} \tag{2.2}$$

with the $(n^{(i)} - 1) \times (n^{(i)} - 1)$ identity matrix $I_{n^{(i)}-1}$ and the Schur-complement

$$K_{\mathrm{S}}^{(i)} = \bar{K}^{(i)} - l_i d_i^{-1} r_i. \tag{2.3}$$

In order to limit the fill-in, the Schur-complement is approximated by a matrix with less non-zero entries. The pattern of the Schur-complement approximation is controlled by a set of *parent indices* P_i containing usually two indices. The construction of these sets is discussed in Sect. 4.

Definition 2.1. *Let a test vector* $t^{(i)} \in \mathbb{R}^{n^{(i)}}$ *and a set of* parent indices P_i *with*

$$\sum_{z \in P_i} {t_{z+1}^{(i)}}^2 > 0 \tag{2.4}$$

be given. Then, the Schur-complement $K_{\mathrm{S}}^{(i)}$ *in (2.3) is approximated by*

$$K^{(i+1)} = \bar{K}^{(i)} - l_i d_i^{-1} a_i - b_i d_i^{-1} r_i + b_i d_i^{-1} a_i, \tag{2.5}$$

$K^{(i+1)} \in \mathbb{R}^{n^{(i+1)} \times n^{(i+1)}}$, $a_i \in \mathbb{R}^{1 \times n^{(i+1)}}$, $b_i \in \mathbb{R}^{n^{(i+1)} \times 1}$, $n^{(i+1)} = n^{(i)} - 1$. *The vectors* a_i *and* b_i *are defined by*

$$(a_i)_j = \begin{cases} \dfrac{|(r_i)_j| \sum\limits_{z \le n^{(i+1)}} (r_i)_z t_{z+1}^{(i)}}{\sum\limits_{z \in P_i} |(r_i)_z|\, t_{z+1}^{(i)}} & : \quad j \in P_i \ \wedge \sum\limits_{z \in P_i} |(r_i)_z|\, t_{z+1}^{(i)} \ge \delta, \\ \dfrac{t_{j+1}^{(i)} \sum\limits_{z \le n^{(i+1)}} (r_i)_z t_{z+1}^{(i)}}{\sum\limits_{z \in P_i} {t_{z+1}^{(i)}}^2} & : \quad j \in P_i \ \wedge \sum\limits_{z \in P_i} |(r_i)_z|\, t_{z+1}^{(i)} < \delta, \\ 0 & : \quad j \notin P_i, \end{cases} \tag{2.6}$$

$$(b_i)_j = \begin{cases} \dfrac{|(l_i)_j| \sum\limits_{z \le n^{(i+1)}} (l_i)_z t_{z+1}^{(i)}}{\sum\limits_{z \in P_i} |(l_i)_z|\, t_{z+1}^{(i)}} & : \quad j \in P_i \ \wedge \sum\limits_{z \in P_i} |(l_i)_z|\, t_{z+1}^{(i)} \ge \delta, \\ \dfrac{t_{j+1}^{(i)} \sum\limits_{z \le n^{(i+1)}} (l_i)_z t_{z+1}^{(i)}}{\sum\limits_{z \in P_i} {t_{z+1}^{(i)}}^2} & : \quad j \in P_i \ \wedge \sum\limits_{z \in P_i} |(l_i)_z|\, t_{z+1}^{(i)} < \delta, \\ 0 & : \quad j \notin P_i. \end{cases} \tag{2.7}$$

$\delta > 0$ *is a small real number, e.g.* $\delta = 10^{-10}$.
The standard choice for the test vector $t^{(i)}$ is

$$t^{(i)} = (1, 1, ..., 1)^T.$$

The construction scheme guarantees the filter conditions

$$(0, a_i - r_i)\, t^{(i)} = 0, \quad (0, (b_i - l_i)^T)\, t^{(i)} = 0.$$

Hence (see Proposition 3.1)

$$\begin{pmatrix} 0 & 0 \\ 0 & K^{(i+1)} - K_{\mathrm{S}}^{(i)} \end{pmatrix} t^{(i)} = \begin{pmatrix} 0 & 0 \\ 0 & (b_i - l_i) d_i^{-1} (a_i - r_i) \end{pmatrix} t^{(i)} = 0$$

and

$$\begin{pmatrix} 0 & 0 \\ 0 & K^{(i+1)} - K_{\mathrm{S}}^{(i)} \end{pmatrix}^T t^{(i)} = 0.$$

2.2 The Two-Level MLILU Decomposition

Definition 2.1 describes a sequence of matrices $K^{(i)} \in \mathbb{R}^{n^{(i)} \times n^{(i)}}$ of decreasing size $n^{(i+1)} = n^{(i)} - 1$. The matrix $K^{(i+1)}$ is obtained by approximating the Schur-complement of $K^{(i)}$ with (2.5) after the elimination of the first column of $K^{(i)}$. The corresponding sequences d_i, r_i and l_i are defined by (2.1).

For the construction of the two-level MLILU decomposition, we assume a given partitioning of the vector of the unknowns $u \in \mathbb{R}^n$ into F- and C-unknowns, $u_{\mathrm{F}} \in \mathbb{R}^{n_{\mathrm{F}}}$, $u_{\mathrm{C}} \in \mathbb{R}^{n_{\mathrm{C}}}$, $n_{\mathrm{F}} + n_{\mathrm{C}} = n$, respectively. Ordering the F-unknowns first leads to $u^T = (u_{\mathrm{F}}^T, u_{\mathrm{C}}^T)$. Then, the first n_{F} columns of the system matrix K are eliminated according to Definition 2.1. An algorithm for constructing the partition into F- and C-unknowns is discussed in Sect. 4.

Definition 2.2. *The operator $[v]_i$ returns the last i entries of the vector v.*

The construction of each matrix $K^{(i)}$, $i > 1$ requires a test vector $t^{(i)}$ (see Definition 2.1). For the two-level MLILU decomposition, these test vectors $t^{(i)}$ are determined by the scheme

$$t^{(i)} = [t]_{n^{(i)}} \in \mathbb{R}^{n^{(i)}}, \quad t \in \mathbb{R}^n. \tag{2.8}$$

Definition 2.3. *(two-level MLILU) Let a matrix K, a test vector $t \in \mathbb{R}^n$ and sets of parents indices P_i, $i = 1, \ldots, n_{\mathrm{F}}$, satisfying (2.4) be given. Then, if $d_i \neq 0$, $i = 1, \ldots, n_{\mathrm{F}}$, the two-level MLILU decomposition M_{TL} is defined by*

$$M_{\mathrm{TL}} = L^{(1)} \cdots L^{(n_{\mathrm{F}})} \begin{pmatrix} I_{n_{\mathrm{F}}} & 0 \\ 0 & K^{(n_{\mathrm{F}}+1)} \end{pmatrix} U^{(n_{\mathrm{F}})} \cdots U^{(1)} \tag{2.9}$$

with lower triangular matrices $L^{(i)}$ and upper triangular matrices $U^{(i)}$, $i = 1, \ldots, n_{\mathrm{F}}$,

$$L^{(i)} = \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & l_i\, d_i^{-1} & I_{n-i} \end{pmatrix}, \qquad U^{(i)} = \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & d_i & r_i \\ 0 & 0 & I_{n-i} \end{pmatrix},$$

where I_i denotes the $i \times i$ identity matrix. The sequences $K^{(i)}$, d_i, r_i, l_i and $t^{(i)}$ are determined by Definition 2.1, the partitioning (2.1) and (2.8) starting at $K^{(1)} = K$.

The two-level MLILU decomposition represents a successive elimination of the first n_{F} columns of K, where all Schur-complements are approximated by (2.5). Matrices which allow a well defined MLILU decomposition are analyzed in Sect. 3.

2.3 The ℓ-Level MLILU Decomposition

Once the two-level decomposition is defined, the construction of the multilevel decomposition is straightforward. The approximate Schur-complement $K^{(n_{\mathrm{F}}+1)}$ in Definition 2.3 is considered as a new matrix and the vector u_{C} is considered as a new vector $u_2 \in \mathbb{R}^{n_2}$ with a new partition into m_2 F- and $n_3 = n_2 - m_2$ C-unknowns. Then, a two-level MLILU decomposition of the matrix K_2 can be calculated.

The recursive ordering of the unknowns leads to a block vector

$$u = (x_1, x_2, \ldots, x_\ell)^T, \quad x_i \in \mathbb{R}^{m_i}, \quad n = \sum_{i \le \ell} m_i$$

where ℓ is the number of levels. Thus, for each level i, u_i

$$u_i = (x_i, \ldots, x_\ell)^T, \quad u_i \in \mathbb{R}^{n_i}, \quad n_i = \sum_{i \le j \le \ell} m_j$$

represents all unknowns, while x_i denotes the F-unknowns. Additionally, m_i and n_i are the number of F-unknowns and the total number of unknowns on the level i respectively. The numbers n_i are supposed to decline by a factor between $1/2$ and $1/4$. This factor will not be constant, but nevertheless the number of levels ℓ will be proportional to the logarithm of the number of unknowns n.

DEFINITION 2.4. *(MLILU) Let a matrix $K \in \mathbb{R}^{n \times n}$, a test vector $t \in \mathbb{R}^n$ and sets of parents indices P_i, $i = 1, \ldots, n - n_\ell$, satisfying (2.4) be given. M_i denotes the two-level MLILU decomposition*

$$M_i = L_i^{(1)} \cdots L_i^{(m_i)} \begin{pmatrix} I_{m_i} & 0 \\ 0 & K_{i+1} \end{pmatrix} U_i^{(m_i)} \cdots U_i^{(1)},$$

of K_i with respect to the test vector $t_i = [t]_{n_i}$, where $K_1 = K$ ($K_i, M_i, U_i^{(j)}, L_i^{(j)} \in \mathbb{R}^{n_i \times n_i}$) . Then, the ℓ-level MLILU decomposition M of K is defined by

$$M = \mathcal{L}_1 \cdots \mathcal{L}_{\ell-1} \begin{pmatrix} I_{n-n_\ell} & 0 \\ 0 & K_\ell \end{pmatrix} \mathcal{U}_{\ell-1} \cdots \mathcal{U}_1, \tag{2.10}$$

where

$$\begin{aligned} \mathcal{L}_i &= \begin{pmatrix} I_{n-n_i} & 0 \\ 0 & L_i \end{pmatrix}, \quad L_i = L_i^{(1)} \cdots L_i^{(n_i)}, \\ \mathcal{U}_i &= \begin{pmatrix} I_{n-n_i} & 0 \\ 0 & U_i \end{pmatrix}, \quad U_i = U_i^{(1)} \cdots U_i^{(n_i)}. \end{aligned}$$

Note that the ℓ-level MLILU decomposition and a two-level MLILU decomposition with $n_C = n_l$ and the same ordering are absolutely equivalent. However, in the following section, we will add smoothing steps on these levels, which is of course not possible without distinct levels.

2.4 The Algorithm

As explained in Sect. 3, the MLILU decomposition M approximates K well for vectors which are in some sense close to the test vector t. On the other hand, the approximation might be poor for other vectors. Therefore, the construction of a second decomposition or the use of a second iterative scheme which takes care of these vectors will be necessary. Hence, a reasonable strategy is to construct an MLILU decomposition with respect to a smooth test vector which handles the smooth error components, and to use smoothing steps, e.g. Gauß-Seidel method, on each level to damp the high frequency error components.

ALGORITHM 2.5. *Assume the matrices U_i, L_i and K_i have been computed by an ℓ-level MLILU decomposition and smoothers S_i are defined. Then, the following function MLILU$(1, u, f)$ calculates one step of the corresponding MLILU method.*

MLILU(i, u_i, f_i)
{
  if$(i == \ell)$ $u_i = K_i^{-1} f_i$;
  else
  {
    $u_i = S_i^{\nu_1}(u_i, f_i)$;
    $d_i = f_i - K_i u_i$;
    $d_i = L_i^{-1} d_i$;
    $d_{i+1} = [d_i]_{n_{i+1}}$;
    $v_{i+1} = 0$;
    for$(j = 0; j < \gamma; j = j + 1)$ **MLILU**$(i + 1, v_{i+1}, d_{i+1})$;
    $[d_i]_{n_{i+1}} = v_{i+1}$;
    $u_i = u_i + U_i^{-1} d_i$;
    $u_i = S_i^{\nu_2}(u_i, f_i)$;
  }
}

$u_i, d_i, f_i, v_i \in \mathbb{R}^{n_i}$.

We emphasize that the entire MLILU decomposition M, including the matrices L_i and U_i in Algorithm 2.5, can be stored as one (sparse) $n \times n$ matrix, as in the case of standard LU decomposition and classical ILU decomposition. Only the matrices K_i, and possibly decompositions for the smoothers S_i, require additional storage. As we will show in Sect. 5, Algorithm 2.5 works well with a simple Gauß-Seidel smoother, so extra memory for a more sophisticated smoother does not seem to be justified at this point.

3 Theoretical Results

In the first part of this section, we prove the existence of the MLILU decomposition and the convergence of Algorithm 2.5 without smoothing steps for symmetric and positive definite matrices. In the rest of the section, we discuss the filter property of the MLILU decomposition and Algorithm 2.5.

3.1 The Error Matrix

The error matrix $N = M - K$ is the difference between the system matrix K and the MLILU decomposition M. As a first step, we investigate the error matrix after one elimination step, essentially the difference between the Schur-complement and the approximation introduced in Definition 2.1. Next, we show that the error matrix of the entire MLILU decomposition is just the sum of the error matrices for each individual elimination step.

PROPOSITION 3.1. *Let $M^{(i)}$ denote the factorization*

$$M^{(i)} = \begin{pmatrix} 1 & 0 \\ l_i d_i^{-1} & I_{n^{(i)}-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & K^{(i+1)} \end{pmatrix} \begin{pmatrix} d_i & r_i \\ 0 & I_{n^{(i)}-1} \end{pmatrix} \tag{3.1}$$

of $K^{(i)}$ in (2.1) as described in Definition 2.1. The comparison with the exact factorization (2.2) yields

$$\begin{aligned} N^{(i)} &= M^{(i)} - K^{(i)} \\ &= \begin{pmatrix} 1 & 0 \\ l_i d_i^{-1} & I_{n^{(i)}-1} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & K^{(i+1)} - K_{\mathrm{S}}^{(i)} \end{pmatrix} \begin{pmatrix} d_i & r_i \\ 0 & I_{n^{(i)}-1} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ l_i d_i^{-1} & I_{n^{(i)}-1} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & (b_i - l_i) d_i^{-1} (a_i - r_i) \end{pmatrix} \begin{pmatrix} d_i & r_i \\ 0 & I_{n^{(i)}-1} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ 0 & (b_i - l_i) d_i^{-1} (a_i - r_i) \end{pmatrix}. \end{aligned} \tag{3.2}$$

LEMMA 3.2. *The error matrix of the two-level MLILU decomposition is given by*

$$N_{\mathrm{TL}} = M_{\mathrm{TL}} - K = \sum_{i \le n_{\mathrm{F}}} \begin{pmatrix} 0_{i-1} & 0 \\ 0 & N^{(i)} \end{pmatrix}, \tag{3.3}$$

where 0_i is an $i \times i$ matrix with $(0_i)_{r,s} = 0 \ \forall r, s$.

Proof. Note that

$$\begin{pmatrix} 0_i & 0 \\ 0 & Z \end{pmatrix} \begin{pmatrix} * & * \\ 0 & I_j \end{pmatrix} = \begin{pmatrix} 0_i & 0 \\ 0 & Z \end{pmatrix}$$

holds for a $(n-i) \times (n-i)$ matrix Z and $j \ge n - i$. Hence,

$$\begin{aligned} M_{\mathrm{TL}} &= L^{(1)} \cdots L^{(n_{\mathrm{F}})} \begin{pmatrix} I_{n_{\mathrm{F}}} & 0 \\ 0 & K^{(n_{\mathrm{F}}+1)} \end{pmatrix} U^{(n_{\mathrm{F}})} \cdots U^{(1)} \\ &= L^{(1)} \cdots L^{(n_{\mathrm{F}}-1)} \begin{pmatrix} I_{n_{\mathrm{F}}-1} & 0 \\ 0 & M^{(n_{\mathrm{F}})} \end{pmatrix} U^{(n_{\mathrm{F}}-1)} \cdots U^{(1)} \\ &= L^{(1)} \cdots L^{(n_{\mathrm{F}}-1)} \begin{pmatrix} I_{n_{\mathrm{F}}-1} & 0 \\ 0 & K^{(n_{\mathrm{F}})} + N^{(n_{\mathrm{F}})} \end{pmatrix} U^{(n_{\mathrm{F}}-1)} \cdots U^{(1)} \\ &= L^{(1)} \cdots L^{(n_{\mathrm{F}}-1)} \begin{pmatrix} I_{n_{\mathrm{F}}-1} & 0 \\ 0 & K^{(n_{\mathrm{F}})} \end{pmatrix} U^{(n_{\mathrm{F}}-1)} \cdots U^{(1)} + \begin{pmatrix} 0_{n_{\mathrm{F}}-1} & 0 \\ 0 & N^{(n_{\mathrm{F}})} \end{pmatrix} \\ &\qquad \vdots \\ &= L^{(1)} \begin{pmatrix} 1 & 0 \\ 0 & K^{(2)} \end{pmatrix} U^{(1)} + \sum_{2 \le i \le n_{\mathrm{F}}} \begin{pmatrix} 0_{i-1} & 0 \\ 0 & N^{(i)} \end{pmatrix} \\ &= K^{(1)} + N^{(1)} + \sum_{2 \le i \le n_{\mathrm{F}}} \begin{pmatrix} 0_{i-1} & 0 \\ 0 & N^{(i)} \end{pmatrix}. \end{aligned}$$

Finally, $K^{(1)} = K$ proves (3.3). □

The same technique as in Lemma 3.2 can be applied for the computation of the error matrix of an ℓ-level MLILU decomposition.

LEMMA 3.3. *Let $N_i = M_i - K_i$ be the error matrix of a two-level MLILU decomposition of the matrix K_i. Then, the error matrix N of an ℓ-level MLILU decomposition*

M of K is given by

$$N = M - K = \sum_{i<\ell} \begin{pmatrix} 0_{n-n_i} & 0 \\ 0 & N_i \end{pmatrix}. \tag{3.4}$$

Proof. Similar to the proof of Lemma 3.2. □

The next lemma shows that if the system matrix K is symmetric and positive definite, all error matrices are symmetric and positive semi-definite. This proves the existence of the MLILU decomposition and the convergence of the corresponding iterative method.

LEMMA 3.4. *Assume K is symmetric and positive definite (s.p.d.). Then, the matrices $K^{(i)}$ and $M^{(i)}$ determined by Definition 2.1, equation (3.1) and $K^{(1)} = K$ are s.p.d.. All error matrices $N^{(i)} = M^{(i)} - K^{(i)}$ are symmetric and positive semi-definite.*

Proof. Note that

$$l_i = r_i^T \quad \Rightarrow \quad b_i = a_i^T$$

(Definition 2.1). Hence,

$$\begin{aligned} N^{(i)} &= \begin{pmatrix} 0 & 0 \\ 0 & (b_i - l_i) d_i^{-1} (a_i - r_i) \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ b_i - l_i & 0 \end{pmatrix} \begin{pmatrix} d_i^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & a_i - r_i \\ 0 & 0 \end{pmatrix} \end{aligned}$$

is obviously symmetric and positive semi-definite if $d_i > 0$.

Induction:
$i = 1$: Since $K^{(1)} = K$ is s.p.d., $d_1 > 0$.
$i \to i + 1$:

$$\begin{aligned} K^{(i+1)} &= \bar{K}^{(i)} - l_i d_i^{-1} a_i - b_i d_i^{-1} r_i + b_i d_i^{-1} a_i \\ &= K_{\mathrm{S}}^{(i)} + (b_i - l_i) d_i^{-1} (a_i - r_i). \end{aligned}$$

Because $K^{(i)}$ is s.p.d., the Schur-complement $K_{\mathrm{S}}^{(i)}$ is s.p.d. and $d_i > 0$. Therefore, $K^{(i+1)}$ is s.p.d.. Hence, $d_{i+1} > 0$. □

COROLLARY 3.5. *The MLILU decomposition M of a s.p.d. matrix K exists and is s.p.d.. The error matrix $N = M - K$ is symmetric and positive semi-definite.*

THEOREM 3.6. *Let K be s.p.d.. Then, the energy norm of the iteration matrix $I_n - M^{-1}K$, where M is the MLILU decomposition of K, is smaller than one*

$$\|I_n - M^{-1}K\|_K < 1.$$

Thus, the iterative method

$$u^{(i+1)} = u^{(i)} + M^{-1}(f - K\,u^{(i)})$$

converges monotonically with respect to the energy norm towards the solution of the linear system $K\,u = f$.

Proof. The proof is based on the positive semi-definiteness of the error matrix. A detailed proof can be found in [13] (Remark 4.8.3), [23] or [24]. □

3.2 The Iteration Matrix

In this section, the iteration matrix of the MLILU algorithm (Algorithm 2.5) is analyzed.

PROPOSITION 3.7. *The iteration matrix $T_{i,TL}$ of Algorithm 2.5 applied as a two-level method on level i is given by*

$$T_{i,TL} = S_i^{\nu_2} \left\{ I_{n_i} - U_i^{-1} \begin{pmatrix} I_{m_i} & 0 \\ 0 & K_{i+1}^{-1} \end{pmatrix} L_i^{-1} K_i \right\} S_i^{\nu_1}.$$

For the multilevel method, K_{i+1}^{-1} is approximated by γ steps of the same method on the level $i+1$. In particular, the solution w_{i+1} of

$$K_{i+1} w_{i+1} = d_{i+1}$$

is approximated by $v_{i+1} = w_{i+1}^{(\gamma)}$, where

$$w_{i+1} - v_{i+1} = w_{i+1} - w_{i+1}^{(\gamma)} = T_{i+1}^{\gamma}(w_{i+1} - w_{i+1}^{(0)}) = T_{i+1}^{\gamma} w_{i+1},$$

because $w_{i+1}^{(0)} = 0$. T_{i+1} denotes the iteration matrix of Algorithm 2.5 starting on level $i+1$. Therefore,

$$v_{i+1} = (I_{n_{i+1}} - T_{i+1}^{\gamma})\, w_{i+1} = (I_{n_{i+1}} - T_{i+1}^{\gamma})\, K_{i+1}^{-1} d_{i+1}.$$

This leads us to the following remark.

REMARK 3.8. *The iteration matrix T_i of Algorithm 2.5 applied on level i is given by*

$$T_i = S_i^{\nu_2} \left\{ I_{n_i} - U_i^{-1} \begin{pmatrix} I_{m_i} & 0 \\ 0 & \tilde{M}_{i+1}^{-1} \end{pmatrix} L_i^{-1} K_i \right\} S_i^{\nu_1},$$

where

$$\begin{aligned} \tilde{M}_{i+1}^{-1} &= (I_{n_{i+1}} - T_{i+1}^{\gamma}) K_{i+1}^{-1}, \quad i < \ell - 1, \\ \tilde{M}_{\ell}^{-1} &= K_{\ell}^{-1}. \end{aligned}$$

3.3 The Filter Property

A characteristic property of the MLILU decomposition is the filter property $Mt = Kt$, where M is the MLILU decomposition, K the system matrix and t is the test vector. We begin our proof with the following lemma.

LEMMA 3.9. *With the same notation as in Proposition 3.1 and Definition 2.1,*

$$N^{(i)}\, t^{(i)} = 0, \quad {N^{(i)}}^T\, t^{(i)} = 0$$

holds.

Proof. Using Definition 2.1,

$$(0, a_i - r_i)\, t^{(i)} = 0, \quad (0, (b_i - l_i)^T)\, t^{(i)} = 0$$

can be easily verified. Thus, (3.2) proves the lemma. □

LEMMA 3.10. *The error matrices N_{TL} (3.3) and N (3.4) satisfy*

$$\begin{aligned} N_{\mathrm{TL}}\, t &= 0, \quad N_{\mathrm{TL}}^T\, t = 0, \\ N\, t &= 0, \quad N^T\, t = 0, \end{aligned}$$

where t represents the test vector in Definition 2.3 and Definition 2.4.

Proof. We prove the result for the two-level decomposition first.

$$N_{\mathrm{TL}}\, t = \sum_{i \le n_{\mathrm{F}}} \begin{pmatrix} 0_{i-1} & 0 \\ 0 & N^{(i)} \end{pmatrix} t = \begin{pmatrix} \vec{0}_{i-1} \\ \sum\limits_{i \le n_{\mathrm{F}}} N^{(i)} [t]_{n^{(i)}} \end{pmatrix} = \begin{pmatrix} \vec{0}_{i-1} \\ \sum\limits_{i \le n_{\mathrm{F}}} N^{(i)} t^{(i)} \end{pmatrix} = 0,$$

where $\vec{0}_i = (0, \ldots, 0)^T \in \mathbb{R}^i$. Hence,

$$N_{\mathrm{TL}}^T\, t = \begin{pmatrix} \vec{0}_{i-1} \\ \sum\limits_{i \le n_{\mathrm{F}}} {N^{(i)}}^T t^{(i)} \end{pmatrix} = 0.$$

Since N_i is the error matrix of a two-level MLILU decomposition with respect to the test vector $t_i = [t]_{n_i}$,

$$N\, t = \sum_{i<\ell} \begin{pmatrix} 0_{n-n_i} & 0 \\ 0 & N_i \end{pmatrix} t = \begin{pmatrix} \vec{0}_{n-n_i} \\ \sum\limits_{i<\ell} N_i [t]_{n_i} \end{pmatrix} = \begin{pmatrix} \vec{0}_{n-n_i} \\ \sum\limits_{i<\ell} N_i\, t_i \end{pmatrix} = 0$$

and

$$N^T\, t = \begin{pmatrix} \vec{0}_{n-n_i} \\ \sum\limits_{i<\ell} N_i^T\, t_i \end{pmatrix} = 0$$

holds. □

Lemma 3.10 shows the filter property

$$M\, t = K\, t, \quad (I_n - M^{-1} K)\, t = 0$$

for the MLILU decomposition M. The following remark explains that Algorithm 2.5 with $\nu_1 = 0$, $\nu_2 \ge 0$ and $\gamma \ge 1$ preserves the filter property.

REMARK 3.11. *Let T_i and t_i be defined by Remark 3.8 and Definition 2.4 respectively. Then, if $\nu_1 = 0$, $\nu_2 \ge 0$ and $\gamma \ge 1$,*

$$T_i\, t_i = 0 \quad \forall i < \ell.$$

Proof.
Induction:
$i = \ell - 1$:

$$\begin{aligned} T_{\ell-1} &= S_{\ell-1}^{\nu_2} \left(I_{n_{\ell-1}} - U_{\ell-1}^{-1} \begin{pmatrix} I_{m_{\ell-1}} & 0 \\ 0 & K_\ell^{-1} \end{pmatrix} L_{\ell-1}^{-1}\, K_{\ell-1} \right) \\ &= S_{\ell-1}^{\nu_2} \left(I_{n_{\ell-1}} - M_{\ell-1}^{-1} K_{\ell-1} \right). \end{aligned}$$

Since $M_{\ell-1}$ is the MLILU decomposition of $K_{\ell-1}$ with respect to $t_{\ell-1}$

$$T_{\ell-1}\, t_{\ell-1} = 0.$$

$i+1 \to i$: Lemma 3.10 yields

$$M_i^{-1} K_i\, t_i = t_i.$$

Hence,

$$\begin{aligned}
U_i^{-1} \begin{pmatrix} I_{m_i} & 0 \\ 0 & K_{i+1}^{-1} \end{pmatrix} L_i^{-1}\, K_i\, t_i &= t_i, \\
\begin{pmatrix} I_{m_i} & 0 \\ 0 & K_{i+1}^{-1} \end{pmatrix} L_i^{-1}\, K_i\, t_i &= U_i\, t_i, \\
\begin{pmatrix} I_{m_i} & 0 \\ 0 & I_{n_{i+1}} - T_{i+1}^{\gamma} \end{pmatrix} \begin{pmatrix} I_{m_i} & 0 \\ 0 & K_{i+1}^{-1} \end{pmatrix} L_i^{-1}\, K_i\, t_i &= \begin{pmatrix} I_{m_i} & 0 \\ 0 & I_{n_{i+1}} - T_{i+1}^{\gamma} \end{pmatrix} U_i\, t_i, \\
\begin{pmatrix} I_{m_i} & 0 \\ 0 & \tilde{M}_{i+1}^{-1} \end{pmatrix} L_i^{-1}\, K_i\, t_i &= \begin{pmatrix} I_{m_i} & 0 \\ 0 & I_{n_{i+1}} - T_{i+1}^{\gamma} \end{pmatrix} U_i\, t_i.
\end{aligned}$$

Note that

$$U_i = \begin{pmatrix} * & * \\ 0 & I_{n_{i+1}} \end{pmatrix}$$

and

$$T_{i+1}\, t_{i+1} = 0.$$

Therefore,

$$\begin{pmatrix} I_{m_i} & 0 \\ 0 & I_{n_{i+1}} - T_{i+1}^{\gamma} \end{pmatrix} U_i\, t_i = U_i\, t_i$$

and

$$U_i^{-1} \begin{pmatrix} I_{m_i} & 0 \\ 0 & \tilde{M}_{i+1}^{-1} \end{pmatrix} L_i^{-1}\, K_i\, t_i = t_i$$

which proves

$$T_i\, t_i = 0.$$

□

4 The Labeling

For each level, i.e. for all systems $K_i\, u_i = f_i$, we must construct a partition into F-unknowns and C-unknowns. Additionally, a set of parent indices P_i must be assigned to each F-unknown. In order to simplify the notation, we omit the level index in this section. The labeling and the parent node assignment is performed for all levels by the same algorithm. This algorithm represents a compromise between a good approximation of the vectors r_i and l_i by a_i and b_i in Definition 2.1 and a small

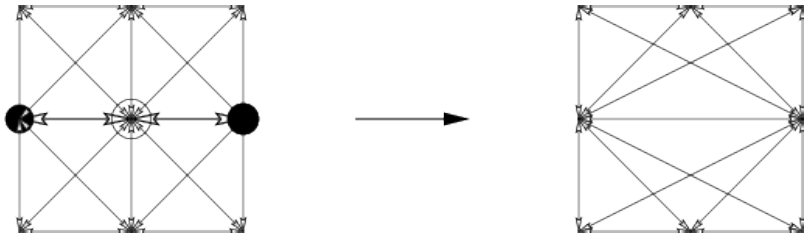

FIG. 1. *New edges after the elimination of the node in the center.*

amount of fill-in. The optimal approximation ($a_i = r_i$, $b_i = l_i$, exact Gaussian Elimination) produces too much fill-in. At the other extreme, the choice $a_i = b_i = 0$ avoids all fill-in but yields the slowly converging symmetric Gauß-Seidel method.

Since we want a robust algorithm, we label an unknown as an F-unknown only if an appropriate set of parent indices can be assigned. Besides this condition, all other elements of the algorithm are motivated only by the goal of minimizing the fill-in and the number of C-unknowns.

4.1 The Graph

The labeling algorithm is based on the graph $G(N, E)$ of the matrix K which consists of a set of nodes N and a set of edges E. Each diagonal entry $k_{i,i}$ and therefore each index i (and each unknown) corresponds to a node in N. Two nodes $r, s \in N$ are connected via an edge $e_{r,s} \in E$ if either $k_{i(r),i(s)} \neq 0$ or $k_{i(s),i(r)} \neq 0$, where $i(r), i(s)$ are the indices corresponding to r, s. Since we are dealing in this section with the graph, in some expressions we replace index with node. For instance, the set of parent indices P_i is now called set of parent nodes. Since both sets are equivalent after the introduction of an ordering, we denote both sets with P_i.

The following proposition summarizes the influence of the elimination procedure and the approximation of the Schur-complement on the graph.

PROPOSITION 4.1. *Assume B_i is the set of neighbor nodes of the node i and P_i is the set of parent nodes. In order to fit into the notation of Sect. 2, we assume without loss of generality, that the node i corresponds to the first column in K. Then, all possible new edges $e_{r,s}$ produced by the approximation of the Schur-complement according to Definition 2.1 satisfy*

$$r \in P_i \ \wedge \ s \in B_i \quad or \quad s \in P_i \ \wedge \ r \in B_i \quad or \quad r \in P_i \ \wedge \ s \in P_i.$$

In other words, the approximation scheme (Definition 2.1) connects all neighbor nodes of the eliminated node with all parent nodes. Fig. 1 shows the ten ($2 \cdot 5$!) new edges after the elimination of the node marked by a circle. The filled dots indicate the parents nodes.

The set of neighbor nodes is not yet defined. One obvious definition would be

$$\overline{B}_i = \{j \in N \mid e_{i,j} \text{ exists}\}. \tag{4.1}$$

Of course, the graph changes after each elimination of a column (node). The elimination of the nodes produces a sequence of graphs which we call *virtual graphs*. Each

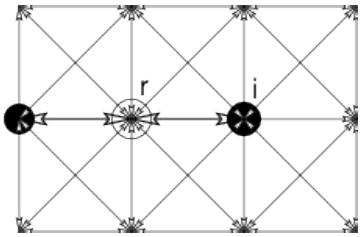

FIG. 2. *All nodes are neighbors of the node* i.

virtual graph arises from the previous virtual graph by adding all new edges produced by the elimination process (Definition 2.1) of a node. The initial virtual graph is the initial graph of the matrix K. Considering the current virtual graph, the labeling algorithm minimizes the number of new edges in the next virtual graph.

However, we do not want to store or update the virtual graphs for several reasons. First, we want to avoid the allocation of new memory during the labeling process. Second, we want all parent nodes of an F-node to be neighbor nodes in the initial graph.

Similar to the code, we use only the edges of the initial graph of the matrix K for the description of the labeling algorithm. Therefore, we introduce a modified definition of a neighbor node (Definition 4.3).

DEFINITION 4.2. *The graph G of the system matrix K is a set of nodes N and edges E. An edge $e_{r,s}$ between two nodes $r, s \in N$ exists, if and only if*

$$k_{i(r),i(s)} \neq 0 \text{ or } k_{i(s),i(r)} \neq 0,$$

where $i(r)$ returns the matrix index corresponding to the node r.

The set $F \subset N$ contains all nodes labeled as F-node. The set $C \subset N$ contains all nodes labeled as C-node. For the partition in Sect. 2.2, the F-nodes are ordered in the same order as they are added to the set F. The C-nodes are ordered in the reverse order as they are added to the set C.

4.2 The Neighbors

DEFINITION 4.3. *(neighbor node) $j \in N \setminus F$ is a neighbor node of $i \in N \setminus F$, if and only if $j \neq i$ and one of the following statements is true.*

- *$e_{i,j}$ exists.*
- *$e_{i,r}$ exists, $r \in F$ and $j \in P_r$.*
- *$e_{i,r}$ exists, $r \in F$, $i \in P_r$ and j is neighbor of r.*

The set of neighbor nodes of i is denoted by B_i, the number of neighbor nodes by nb_i.

The labeling algorithm insures that the recursive definition of a neighbor node does not cause any problem, because a node in F cannot serve as a parent node. Therefore, the depth of the recursion is never larger than one. In the example case shown in Fig. 2, the node $\mathsf{i} \in P_{\mathsf{r}}$ is a parent node of the node $\mathsf{r} \in F$. Thus, according to Definition 4.3, all nodes in Fig. 2 aside from the node r are neighbors of the node i.

REMARK 4.4. *The definition of a neighbor node according to Definition 4.3 is equivalent to (4.1) if $e_{i,j}$ in (4.1) refers to an edge in the current virtual graph.*

The following lemma predicts the number of fill-in edges produced by the approximation scheme. This number is equivalent to the number of new edges in the next virtual graph and to the number of new neighbors according to Definition 4.3.

LEMMA 4.5. *Let* $P_i \subset B_i$,

$$np_i = \textit{number of parent nodes of } i,$$
$$ncn(i,j) = \textit{number of common neighbors of } j \textit{ and } i,$$
$$\zeta(r,s) = \begin{cases} 0 & : \textit{ if } e_{r,s} \textit{ exists,} \\ 1 & : \textit{ otherwise.} \end{cases}$$

Then, the maximal number $\eta_i(P_i)$ *of new edges in the next virtual graph is given by*

$$\eta_i(P_i) = 2\,np_i \cdot (nb_i - 1) - 2 \sum_{j \in P_i} ncn(i,j) - 2 \sum_{r,s \in P_i} \zeta(r,s).$$

Proof.
$2\,np_i \cdot (nb_i - 1)$ describes the number of possible new edges in the next virtual graph analyzed by Proposition 4.1. $2 \sum_{j \in P_i} ncn(i,j)$ is the number of already existing edges. If r and s are not connected parent indices, the first term counts the new edges twice. This is compensated by the third term. □

PROPOSITION 4.6. *Assume all possible fill-in edges are in fact stored. Then, once the F-, C-unknowns and the parent indices are selected, the pattern (the graph) of the approximated Schur-complement* $K^{(n_F+1)}$ *after the elimination of the first* n_F *columns of* K *does not depend on the ordering of the F-unknowns.*

4.3 The Algorithm

The first step of the labeling algorithm is the construction of the graph G. Each edge gets a weight depending on the value of the corresponding matrix entry. This is done by the following function.

```
Init_Graph
{
    for((r, s) ∈ N × N)
    {
        create e_{r,s} and e_{s,r} if k_{i(r),i(s)} ≠ 0 or k_{i(s),i(r)} ≠ 0;
        e_{r,s} = { 1 : k_{i(r),i(s)} ≥ σ max_j(k_{i(r),j});
                  { 0 : otherwise;
    }
}
```

The parameter σ is the only parameter in the labeling and parent assignment algorithm. In this paper, only two values $\sigma = 1/2$ and $\sigma = 1/4$ are used.

The main idea of the labeling algorithm is to mark for each node those nodes which would be the optimal parent nodes. The optimal parent nodes of a node i are strongly connected nodes j, i.e. $e_{i,j} > 0$, which minimize the number of new matrix entries (the number of new edges in the next virtual graph). Then, we label the node which is an optimal parent for the largest number of nodes as C-node. After that, we update the optimal parent marks and select the next C-node which has to satisfy the same condition. A node is labeled as F-node as soon as it has two strong neighbors which

are labeled as C-node. Those nodes are the parent nodes. The following functions describe this process in detail.

The core of the algorithm and the only complex part is the selection of the optimal parent nodes. Since we want to minimize the number of new edges in the next virtual graph, we need to compute the number of new edges $\eta_i(p_1, p_2)$ for each possible pair of parent nodes. This is done by Lemma 4.5. Note that $\eta_i(p_1, p_2)$ can be evaluated fast, using flags for the neighbors of the node i. Essentially, only the number of common neighbors $ncn(i, j)$ has to be computed for each possible parent node.

```
Mark_Optimal_Parents(i)
{
        W_i = {j ∈ N \ F | e_{i,j} > 0};

        /* reset marks */
        for(j ∈ W_i) e_{i,j} = 1 ;

        /* minimize new edges */
        Q_i = {(r,s) ∈ W_i × W_i | η_i(r,s) = min_{(l,j)∈W_i×W_i} (η_i(l,j)) };

        /* weakest connection */
        P_i = {(r,s) ∈ Q_i | ξ_i(r,s) = min_{j with (r,j)∈Q_i} (ξ_i(r,j)) };

        /* mark edges */
        if((r,s) ∈ P_i exists with r ∈ C ∨ s ∈ C)
        {
                for((r,s) ∈ P_i)
                {
                        if(r ∈ C ∨ s ∈ C) e_{i,s} = e_{i,r} = 250 + 2 nb_i − η_i(r,s);
                }
        }
        else
        {
                for((r,s) ∈ P_i) e_{i,s} = e_{i,r} = 100 + 2 nb_i − η_i(r,s);
        }
}
```

In most cases, the pair of parent nodes minimizing the number of additional edges is not unique. One node can even minimize the number of additional edges with different second parent nodes. In order to reduce the number of of possible parent pairs, we consider in Mark_Optimal_Parents only those nodes as second node which have the weakest connection to the first node. The connection is measured by $\xi_i(l, j)$.

```
ξ_i(l,j)
{
        ξ_i(l,j) = 0;
        if(e_{l,j} > 0) ξ_i(l,j) = ξ_i(l,j) + 16;
        if(e_{l,j} = 0) ξ_i(l,j) = ξ_i(l,j) + 4;
        for(s ∈ B_l)
        {
                if(e_{l,s} > 0 ∧ e_{j,s} > 0) ξ_i(l,j) = ξ_i(l,j) + 4;
                if(e_{l,s} = 0 ∧ e_{j,s} > 0) ξ_i(l,j) = ξ_i(l,j) + 2;
```

```
            if(e_{l,s} > 0  ∧ e_{j,s} = 0) ξ_i(l,j) = ξ_i(l,j) + 2;
            if(e_{l,s} = 0  ∧ e_{j,s} = 0) ξ_i(l,j) = ξ_i(l,j) + 1;
        }
}
```

A modification of the parameters involved in the computation of ξ changes the results only slightly.

In Mark_Optimal_Parents, we mark the edges to possible parent edges with a large number ($100 + 2\,nb_i - \eta_i(r,s)$ or $250 + 2\,nb_i - \eta_i(r,s)$) in order to distinguish them from the (only) strong edges. The actual value is not important, it just has to be much larger than 1. If one of the optimal parent nodes is already a C-node, we mark only these edges and increase the weight by a factor 2.5, because if this node is selected we get an F-node with an optimal pair of parent nodes. Of course, the factor 2.5 is heuristic. As long as this factor is larger than two the results do not change dramatically. The difference between the "eliminated" edges $2\,nb_i$ and the new edges η_i in the next virtual graph is added in order to break ties.

Once the optimal parent nodes are marked, we can select the node which the most unlabeled nodes want to use as a parent node. This node is chosen by the following function.

```
Next_Node
{
        Y = N \ (F ∪ C);
        if (Y = {}) return(0);

        choose n ∈ Y with λ_n = max_{j∈Y} λ_j;
        return(n);
}
```

λ_i is defined by $\lambda_i = \sum_{j\in Y} e_{j,i}, \quad Y = N \setminus (F \cup C)$.

If more than one node with the maximal λ_n exists, the returned node depends on the implementation. Since we expect only a limited number of values for λ_i, which is in particular independent of the number of nodes, the search can be implemented fast, for instance using a couple of lists.

Finally, we need a function which decides if a node is labeled as F-node and which assigns the parent nodes.

```
Check_if_F-node(i)
{
        np = ne = 0;
        for(j with e_{i,j} > 0)
        {
                ne = ne + 1;
                if(j ∈ C) np = np + 1;
        }
        if(np > 1  ∨  (ne = 1  ∧  np = 1))
        {
                F = F ∪ i;
                P_i = {j ∈ C | e_{i,j} > 0};
        }
}
```

A node is selected as F-node, if at least two neighbors with $e_{i,j} > 0$ can be assigned as parent nodes. If the node i has only one neighbor satisfying $e_{i,j} > 0$, this neighbor node is sufficient as parent node. Note that only nodes which are neighbor nodes in the original graph — in contrast to the current virtual graph — can serve as neighbor nodes.

The last nodes returned by the function Next_Node might have $\lambda_n = 0$. Hence, this node cannot be a parent node. On the other hand, this node could not be eliminated so far. In order to get a faster coarsening, we try to eliminate this node by imposing weaker F-node conditions for the nodes with $\lambda_n = 0$. If an appropriate pair of parent nodes cannot be found, the node is labeled as C-node.

```
Check_if_F-_or_C-Node(i)
{
        if(P_i = B_i does not cause additional edges in the next virtual graph)
        {
                F = F ∪ i;
                P_i = B_i;
                return;
        }
        np = 0;
        for(j ∈ C with e_{i,j} ≥ 0)
        {
                if (e_{i,j} > 0) np = np + 1;
                if ((j ∈ P_s ∧ e_{i,s} > 0) for at least 2 different s)
                {
                        np = np + 1;
                        e_{i,j} = 1;
                }
        }
        if (np > 1)
        {
                Mark_Optimal_Parents(i);
                F = F ∪ i;
                for(j ∈ C with e_{i,j} > 0)
                        if (j is an optimal parent) P_i = P_i ∪ j;
        }
        else C = C ∪ i;
        return;
}
```

For nodes i with $\lambda_i = 0$, a weak edge $(e_{i,j})$ becomes a strong edge $(e_{i,j} = 1)$, if the node j is a parent node of two strong neighbor nodes already marked as F-node. This reflects that the absolute value of the matrix entry to a node j which is a parent node of a strong neighbor F-node increases after the elimination of the F-node.

After the labeling of all nodes, it might be possible to add parent nodes to some F-nodes without changing the virtual graph after the elimination of all F-nodes. This is done by the next function.

```
Free_Parents
{
        for(i ∈ F)
```

```
    {
        for(j ∈ B_i \ P_i)
        {
            if (j ∈ P_i does not cause additional edges in the virtual graph
              after the elimination of all F-nodes)
                P_i = P_i ∪ j;
        }
    }
}
```

Finally, we combine these functions to the labeling and parent node assignment algorithm.

ALGORITHM 4.7. *The following function labels all nodes of the graph G of the system matrix K either as F-node or as C-node and assigns the parent nodes. The F-nodes are the nodes in the set F, the C-nodes are the nodes in the set C. For the partition in Sect. 2.2 the nodes are ordered as explained in Definition 4.2. The sets of parent indices P_i are obtained by replacing the parent nodes with the corresponding indices.*

```
Mark_Nodes
{
    F = C = {};
    Init_Graph;
    for(i ∈ N) Mark_Optimal_Parents(i);
    k = Next_Node;
    while(k ≠ 0)
    {
        if(λ_k > 0)
        {
            C = C ∪ k;
            for( j with e_{j,k} > 0 and j ∉ (F ∪ C)) Check_if_F-node(j);
            for( j ∉ (F ∪ C) and optimal parents might have changed)
                Mark_Optimal_Parents(j);
        }
        else Check_if_F_-or_C-node(k);
        k = Next_Node;
    }
    Free_Parents;
}
```

The optimal parents might change for a node when one of the neighbor nodes is labeled as F- or C-node.

4.4 The Complexity

The most expensive procedure inside the while-loop of the Mark_Nodes function (Algorithm 4.7) is Mark_Optimal_Parents(j). Mark_Optimal_Parents(j) can be implemented with a complexity proportional to $nsb_j \cdot nb$, where nsb_j describes the number of strong neighbors ($e_{j,i} > 0$) and nb the average number of neighbors. Mark_Optimal_Parents(j) is called inside the while-loop for all neighbors of a new F- or C-node. Since $O(n_l)$ nodes are marked during the while-loop, the complexity of the while-loop is proportional to $n_l \cdot nb^2 \cdot nsb_j$, where n_l denotes the number of

nodes in the graph on level l. The complexity of Init_Graph is $O(n_l\, nb)$, the complexity of Free_Parents is $O(n_l\, nb^2)$. The overall complexity of Mark_Nodes is therefore $O(n_l\, nb^2\, nsb)$ with the average number of strong neighbors nsb.

On each level, new matrix entries are produced during the elimination process. Since new matrix entries are only generated between the neighbors of an F-node and the corresponding parent nodes the number of entries in L_l and U_l (see Definition 2.4) is proportional to $n_l \cdot nb + n_{l+1} \cdot nb^2$, where $n_{l+1} \cdot nb^2$ describes the additional edges. Nevertheless, the average stencil size $(nb + 1)$ of the matrix K_{l+1} and the matrix K_l are supposed to be similar. If we assume $n_{l+1} < \varrho\, n_l$, $\varrho < 1$, the complexity of one MLILU V-cycle ($\gamma = 1$) iteration step is $O(n_1\, nb^2)$. We cannot prove that the number of iteration steps for a fixed reduction of the residual is independent of the number of unknowns. However, the numerical experiments in Sect. 5 show convergence rates independent of the number of unknowns. In those cases, the total complexity of the MLILU scheme is proportional to the number of unknowns. According to our numerical experiments, the labeling and the construction of the MLILU decomposition is approximately as expensive as ten iteration steps.

5 Numerical Experiments

In this section, the performance of the MLILU algorithm (Algorithm 2.5) is demonstrated. For all experiments, only one post-smoothing Gauß-Seidel step is applied ($\nu_1 = 0$, $\nu_2 = 1$). If the value of γ is not specifically indicated, we set $\gamma = 1$. As test vector, $t = (1, 1, \ldots, 1)^T$ is used. All differential equations are discretized with the finite volume method (see [12]) using either piecewise linear or piecewise bilinear basis functions.

We report the average convergence rate of the first n steps

$$k_n = \left(\frac{\|f - K\, u^{(n)}\|}{\|f - K\, u^{(0)}\|} \right)^{1/n}, \quad k_n^n < 10^{-10},$$

necessary for a ten orders of magnitude reduction of the residual. The computing times are measured on a Sun Sparc 20 workstation and include the time for the labeling, the construction of the MLILU decomposition and the solution process.

The implementation of the standard multi-grid method (one pre- and one post-smoothing step with Gauß-Seidel) in an old ug [8] version needs 15 s or 20 s to solve the linear systems in Experiment 5.1 on the finest grid ($h = 1/128$) discretized with piecewise bilinear or linear basis functions respectively. This does not include the time for the computation of the coarse grid matrices.

Aside from Experiments 5.1, 5.2 and 5.7, the performance of the standard multi-grid method is poor. However, using special smoothers, special ordering strategies or conjugate gradient acceleration might improve the convergence.

5.1 Poisson Equation

The first experiment analyzes how the convergence rates depend on the number of unknowns.

Experiment 5.1.

$$\Delta\, u = 4 \quad in\ \Omega = (0, 1) \times (0, 1), \tag{5.1}$$

$$u(x,0) = \frac{1.001}{x+0.001}, \quad u(x,1) = 1,$$
$$u(0,y) = \frac{1.001}{y+0.001}, \quad u(1,y) = 1.$$

Table 1 shows average convergence rates and computing times for uniform grids with different mesh sizes h. We used in all cases $\sigma = 1/4$.

h	bilinear basis	bilinear basis	linear basis	linear basis
1/32	$k_8 = 0.047$	$t = 0.93\,\mathrm{s}$	$k_{10} = 0.094$	$t = 0.8\,\mathrm{s}$
1/64	$k_8 = 0.046$	$t = 4.5\,\mathrm{s}$	$k_{10} = 0.098$	$t = 3.7\,\mathrm{s}$
1/128	$k_8 = 0.051$	$t = 18.0\,\mathrm{s}$	$k_{10} = 0.095$	$t = 15.2\,\mathrm{s}$

TABLE 1. *Average convergence rates for Experiment 5.1*

Experiment 5.1 indicates that the MLILU algorithm converges independently of the number of unknowns. The computing times show that the total complexity is proportional to the number of unknowns. The "coarse graphs" on the first levels are very similar to the coarse grids in a standard multi-grid method. This can be verified in Table 2, which shows the number of unknowns and the average stencil size for the MLILU graphs.

level	unknowns (bilin.)	avg. stencil (bilin.)	unknowns (lin.)	avg. stencil (lin.)
0	16129	8.9	16129	6.9
1	3973	8.8	4190	6.9
2	965	8.6	1135	6.9
3	229	8.1	326	6.7
4	53	7.3	124	7.0
5	13	5.9	39	7.5
6	4	4.0	15	6.6
7	2	2.0	4	4.0
8			2	2.0

TABLE 2. *Number of nodes and average stencil size for Experiment 5.1, $h = 1/128$.*

The purpose of the MLILU decomposition is not to solve simple model problems on uniform grids. The next experiment investigates the performance of the algorithm on strongly locally refined grids.

EXPERIMENT 5.2. *After two levels of uniform refinement, equation (5.1) is discretized on grids which are adaptively refined with an a posteriori error indicator. The grids are strongly refined in the lower left corner. Table 3 shows the results for several levels of refinement. Table 4 presents average stencil sizes for the MLILU levels ($\sigma = 1/4$).*

Apart of some fluctuations in the convergence rates for the coarse grids, the computing time is almost proportional to the number of unknowns

5.2 Anisotropic Problems

refinement level	unknowns	convergence	time
2	49	$k_8 = 0.046$	$t = 0.05\,\mathrm{s}$
3	206	$k_8 = 0.052$	$t = 0.18\,\mathrm{s}$
4	372	$k_{12} = 0.14$	$t = 0.38\,\mathrm{s}$
5	526	$k_{11} = 0.12$	$t = 0.56\,\mathrm{s}$
6	658	$k_{11} = 0.11$	$t = 0.72\,\mathrm{s}$
7	779	$k_{12} = 0.13$	$t = 0.91\,\mathrm{s}$
8	880	$k_{12} = 0.13$	$t = 1.10\,\mathrm{s}$
9	973	$k_{12} = 0.13$	$t = 1.25\,\mathrm{s}$
10	1041	$k_{12} = 0.13$	$t = 1.35\,\mathrm{s}$

TABLE 3. *Results for Experiment 5.2.*

unknowns	1041	297	91	29	10	4	2
avg. stencil	8.4	8.8	9.2	8.0	5.4	3.5	2.0

TABLE 4. *MLILU level characteristics for Experiment 5.2.*

EXPERIMENT 5.3. *The results for the anisotropic differential equation*

$$\epsilon_x \tfrac{\partial^2 u}{\partial x^2} + \epsilon_y \tfrac{\partial^2 u}{\partial y^2} = 0 \quad \textit{in } \Omega = (0,1)\times(0,1),$$
$$u(x,y) = \tfrac{x+y}{2} \quad \textit{on } \partial\Omega,$$

where

$$\begin{aligned}
&\epsilon_x = \epsilon, \quad \epsilon_y = 1, && x < 1/2,\ y < 1/2,\\
&\epsilon_x = 1, \quad \epsilon_y = \epsilon, && x \geq 1/2,\ y < 1/2,\\
&\epsilon_x = 1, \quad \epsilon_y = \epsilon, && x < 1/2,\ y \geq 1/2,\\
&\epsilon_x = \epsilon, \quad \epsilon_y = 1, && x \geq 1/2,\ y \geq 1/2,
\end{aligned}$$

which is discretized with linear basis functions on an uniform grid ($h = 1/128$, $\sigma = 1/2$) are presented in Table 5.

ϵ	convergence	time
1	$k_{11} = 0.109$	$t = 16\,\mathrm{s}$
10^{-2}	$k_{15} = 0.207$	$t = 25\,\mathrm{s}$
10^{-4}	$k_{19} = 0.288$	$t = 29\,\mathrm{s}$
10^{-6}	$k_{19} = 0.288$	$t = 29\,\mathrm{s}$

TABLE 5. *Results for Experiment 5.3.*

For Experiment 5.3, the labeling algorithm follows a sort of a semi-coarsening strategy (coarsening in only one direction) which can be seen from Table 6 and Fig. 3.

As an example of an unstructured grid, we compute the electrostatic potential in a drift chamber. In order to resolve small details (the anodes), the coarsest triangulation consists of 112 elements, some with very small angles. Even though the equation is

unknowns	16129	8065	4033	2016	1012	557	260	37	15	6	2
avg. stencil	6.9	8.8	8.9	9.6	9.4	9.3	8.7	6.1	3.8	2.6	2.0

TABLE 6. *MLILU level characteristics for Experiment 5.3.*

only a Poisson equation, the problem is difficult to solve for standard multi-grid methods because triangles with very small angles yield anisotropic stencils.

EXPERIMENT 5.4.

$$\Delta \phi = 0 \quad \textit{in } \mathcal{D},$$

where $\mathcal{D}$ is a complex domain (drift chamber). The boundary conditions are of Neumann and Dirichlet type. A detailed description of this problem can be found in [7] and [23]. The coarsest triangulation consists of 112 triangles, some with very small angles. Table 7 shows convergence rates for several grids, which are obtained from the regular refinement of the coarsest triangulation ($\sigma = 1/2$).

unknowns	convergence	time
890	$k_{15} = 0.210$	$t = 1.0\,\mathrm{s}$
3578	$k_{21} = 0.322$	$t = 5.6\,\mathrm{s}$
14330	$k_{23} = 0.360$	$t = 27\,\mathrm{s}$
14330 ($\gamma = 2$)	$k_{11} = 0.114$	$t = 29\,\mathrm{s}$

TABLE 7. *Results for Experiment 5.4.*

A similar, small dependence of the convergence rates on the number of unknown for the fine grids can be observed for the standard multi-grid method, although the convergence rates are much worse for standard multi-grid.

The stencil sizes and the number of unknowns are presented for the MLILU levels in Table 8.

unknowns	14330	5773	2418	1092	486	214	92	40	15	6
avg. stencil	6.8	8.1	9.0	10.4	12.0	13.3	14.0	11.8	9.2	6.0

TABLE 8. *MLILU level characteristics for Experiment 5.4.*

5.3 Jumping Coefficients

The first experiment in this section represents a standard model problem for interface problems.

EXPERIMENT 5.5.

$$\begin{aligned} \nabla \cdot (D\ \nabla u) &= 0 \quad \textit{in } \Omega = (0,1) \times (0,1), \\ D &= \begin{cases} 1 & : \ \delta < y < 1-\delta \ \wedge \ \delta < x < 1-\delta, \\ \epsilon & : \ \textit{otherwise}, \end{cases} \\ u(x,y) &= \frac{x+y}{2} \quad \textit{on } \partial\Omega. \end{aligned}$$

The convergence results for several values of ϵ and δ can be found in Table 9. The differential equation was discretized with bilinear basis functions on an uniform grid ($h = 1/128$, $\sigma = 1/4$). Since the MLILU graphs are similar to the graphs in Experiment 5.1 except for a perturbation at the interface , we skip these information for this experiment.

ϵ	conv. rate, $\delta = 1/4$	time, $\delta = 1/4$	conv. rate, $\delta = 33/128$	time, $\delta = 33/128$
10^6	$k_9 = 0.072$	$t = 20\,\mathrm{s}$	$k_{16} = 0.24$	$t = 25\,\mathrm{s}$
10^4	$k_9 = 0.072$	$t = 20\,\mathrm{s}$	$k_{16} = 0.24$	$t = 25\,\mathrm{s}$
10^2	$k_9 = 0.072$	$t = 20\,\mathrm{s}$	$k_{16} = 0.24$	$t = 25\,\mathrm{s}$
10^0	$k_8 = 0.051$	$t = 18\,\mathrm{s}$	$k_8 = 0.051$	$t = 18\,\mathrm{s}$
10^{-2}	$k_{10} = 0.095$	$t = 20.5\,\mathrm{s}$	$k_{18} = 0.27$	$t = 27\,\mathrm{s}$
10^{-4}	$k_{10} = 0.097$	$t = 20.5\,\mathrm{s}$	$k_{18} = 0.27$	$t = 27\,\mathrm{s}$
10^{-6}	$k_{10} = 0.099$	$t = 20.5\,\mathrm{s}$	$k_{18} = 0.27$	$t = 27\,\mathrm{s}$

TABLE 9. *Results for Experiment 5.5.*

The convergence behavior of the MLILU method is very robust for Experiment 5.5. Some multi-grid methods have difficulties to solve the linear systems especially if ϵ is very small.

A realistic problem is investigated in the next example. In soil physics, the exact distribution of the conductivity in the soil is rarely known. Therefore, special random generators are often used to produce a conductivity distribution with certain properties. Typical properties are the mean $\overline{\log k_f}$ and the standard deviation $\sigma_{\log k_f}$ of the log-normal distribution as well as the correlation length λ. For a detailed description, we refer to [22]. We compare results of the MLILU algorithm for a short ($\lambda = 0.05$) and a mid range ($\lambda = 0.2$) correlation length. The standard deviation $\sigma_{\log k_f}$ controls the height of the conductivity jumps. Results for $\sigma^2_{\log k_f} = 0.8$ and $\sigma^2_{\log k_f} = 0.4$ are considered corresponding to jumps of about 6 and 4.5 orders of magnitude respectively. For the standard multi-grid method, the most difficult case is the case of a large standard deviation and a short correlation length. The convergence rates even with Galerkin approximation for the coarse grid matrices are around 0.9. For the other problems, the convergence rates are slightly better.

EXPERIMENT 5.6. *The hydraulic conductivity k_f in Darcy's equation for the piezometric head Φ (potential)*

$$\begin{aligned} \nabla \cdot (k_f \nabla \Phi) = 0 \quad &\text{in } \Omega = (0,1) \times (0,1), \\ \Phi(0,y) = 0.001, \quad &\Phi(1,y) = 0, \\ \tfrac{\partial \Phi(x,0)}{\partial y} = 0, \quad &\tfrac{\partial \Phi(x,1)}{\partial y} = 0 \end{aligned}$$

is determined by the random generator fgen92 (see [18] and [21]). For the discretization, bilinear basis function on an uniform mesh ($h = 1/128$) are used. The results are shown in Table 10. Note that σ denotes the parameter of the labeling algorithm, while $\sigma_{\log k_f}$ describes the standard deviation of the conductivity distribution. All results in Table 10 represent the mean of ten different distributions with the same parameters.

Although the convergence rates are much better, due to the higher computational cost, the W-cycle ($\gamma = 2$) in line 2 is only slightly faster than the V-cycle for the

same problems (line 1). Note that, the convergence rates are almost independent of $\sigma^2_{\log k_f}$ (for $\sigma^2_{\log k_f}$ large enough). The number of iterations decreases for increasing correlation lenght which means slower spatial variation of the conductivity.

method	λ	$\sigma^2_{\log k_f}$	avg. # iterations	time
$\gamma = 1$, $\sigma = 1/4$	$\lambda = 0.05$	$\sigma^2_{\log k_f} = 0.8$	25	$t = 38\,\mathrm{s}$
$\gamma = 2$, $\sigma = 1/4$	$\lambda = 0.05$	$\sigma^2_{\log k_f} = 0.8$	13	$t = 34\,\mathrm{s}$
$\gamma = 1$, $\sigma = 1/2$	$\lambda = 0.05$	$\sigma^2_{\log k_f} = 0.8$	22.7	$t = 38\,\mathrm{s}$
$\gamma = 1$, $\sigma = 1/4$	$\lambda = 0.05$	$\sigma^2_{\log k_f} = 0.4$	24.2	$t = 38\,\mathrm{s}$
$\gamma = 1$, $\sigma = 1/2$	$\lambda = 0.05$	$\sigma^2_{\log k_f} = 0.4$	22.2	$t = 34.6\,\mathrm{s}$
$\gamma = 1$, $\sigma = 1/4$	$\lambda = 0.2$	$\sigma^2_{\log k_f} = 0.8$	16.4	$t = 26.5\,\mathrm{s}$
$\gamma = 1$, $\sigma = 1/2$	$\lambda = 0.2$	$\sigma^2_{\log k_f} = 0.8$	16.7	$t = 27.4\,\mathrm{s}$

TABLE 10. *Results for Experiment 5.6.*

The characteristics of the MLILU levels for $\sigma = 1/4$ and $\sigma = 1/2$ are compared in Table 11 for $\lambda = 0.05$ and $\sigma^2_{\log k_f} = 0.8$. In general, a smaller σ produces "coarse" graphs which are more similar to the graph of the system matrix. This trend can be seen in Table 11.

unkn., $\sigma = 1/4$	16383	4255	1314	463	163	63	26	11	3
stencil, $\sigma = 1/4$	8.9	9.4	11.2	13.3	14.3	13.3	14.6	9.1	3.0
unkn., $\sigma = 1/2$	16383	5575	2523	1241	514	194	58	23	7
stencil, $\sigma = 1/2$	8.9	9.9	11.7	13.4	14.8	12.9	15.1	11.4	7.0

TABLE 11. *MLILU level characteristics for Experiment 5.6.*

5.4 Unsymmetric Problems

The first experiment is a convection-diffusion equation. The direction of the convection is uniform but the absolute value of the convection changes within the domain.

EXPERIMENT 5.7.

$$\begin{aligned} \epsilon\,\Delta\,u + c^4\,\cos(\pi\alpha)\tfrac{\partial u}{\partial x} + c^4\,\sin(\pi\alpha)\tfrac{\partial u}{\partial y} = 0 \quad &\textit{in } \Omega = (0,1)\times(0,1), \\ c = 1 - \sin(\pi\,\alpha)\,[2\,(x+1/4) - 1] + 2\,\cos(\pi\,\alpha)(y - 1/4), & \\ u(x,0) = \begin{cases} 1 & : \quad 0.4 < x < 0.6, \\ 0 & : \quad \textit{otherwise}, \end{cases} & \\ u(x,1) = 1, \quad u(1,y) = 0, \quad u(0,y) = 0. & \end{aligned}$$

Table 12 presents convergence rates for a discretization with linear basis functions on an uniform grid ($h = 1/128$) for $\epsilon = 10^{-5}$ and $\sigma = 1/2$.

Due to the discretization, the linear system does not reflect the alignment with the grid lines in the case $\alpha = 1/2$. Table 13 contains the information about the graphs on coarser levels. (We skip the last level which contains five unknowns.

In the next experiment, the direction of the convection changes within the domain. The rotating structure of the convection requires sophisticated ordering strategies for

α	convergence	time
1/6	$k_{12} = 0.125$	$t = 31\,\mathrm{s}$
2/6	$k_{14} = 0.171$	$t = 33\,\mathrm{s}$
3/6	$k_{15} = 0.214$	$t = 33\,\mathrm{s}$
4/6	$k_{12} = 0.130$	$t = 30\,\mathrm{s}$
5/6	$k_{10} = 0.090$	$t = 26\,\mathrm{s}$

TABLE 12. *Results for Experiment 5.7.*

unkn.	16129	8217	4544	2516	1403	787	411	196	99	46	16
stencil	6.9	9.5	11.9	13.3	14.5	15.6	18.0	17.9	17.5	14.5	13.1

TABLE 13. *MLILU level characteristics for Experiment 5.7,* $\alpha = 1/2$.

many solvers. For the MLILU decomposition, the same labeling algorithm (Algorithm 4.7) can be applied.

EXPERIMENT 5.8. *The convection term in the differential equation*

$$\epsilon\,\Delta\,u - \sin(\pi\,x)\cos(\pi y)\tfrac{\partial u}{\partial x} + \sin(\pi\,y)\cos(\pi x)\tfrac{\partial u}{\partial y} = 0 \quad \textit{in } \Omega = (0,1)\times(0,1),$$
$$u(x,y) = \sin(\pi x) + \sin(13\pi x) + \sin(\pi y) + \sin(13\pi y) \quad \textit{on } \partial\Omega$$

simulates a rotating flow. Convergence rates are shown in Table 14. (uniform grid, bilinear basis functions, $h = 1/128$, $\sigma = 1/2$*).*

ϵ	convergence	time
1	$k_9 = 0.066$	$t = 20\,\mathrm{s}$
10^{-2}	$k_{18} = 0.275$	$t = 33\,\mathrm{s}$
10^{-4}	$k_{22} = 0.342$	$t = 39\,\mathrm{s}$
10^{-6}	$k_{21} = 0.333$	$t = 38\,\mathrm{s}$

TABLE 14. *Results for Experiment 5.8.*

Table 15 presents the number of unknowns and the average stencil size for the MLILU levels (see Fig. 4. (We skip the last two levels which contain ten and three unknowns respectively.)

The increasing amount of fill-in for the unsymmetric problems on coarser graphs, could be reduced by lumping (adding) some small off-diagonal matrix entries into the diagonal. This can be done such that the filter property is preserved. However, the investigation of special modifications of the MLILU algorithm is not the purpose of this paper.

5.5 Graphs

Fig. 3 and Fig. 4 show the graph of the forth level (K_4) for Experiment 5.3 and Experiment 5.8 respectively ($\epsilon = 10^{-6}$ for both Experiments). For a better visibility, we applied the MLILU decomposition to a smaller initial linear system ($h = 1/32$). The graphs for the corresponding larger linear systems look similar. The bold lines indicate the connections to parents nodes. The filled dots are the unknowns labeled as C-unknown. The circles mark the F-unknowns.

unkn.	16129	8391	4706	2713	1548	838	455	235	123	62	23
stencil	8.9	9.0	9.8	11.4	12.8	14.0	15.2	16.4	17.1	16.4	18.3

TABLE 15. *MLILU level characteristics for Experiment 5.8,* $\epsilon = 10^{-4}$.

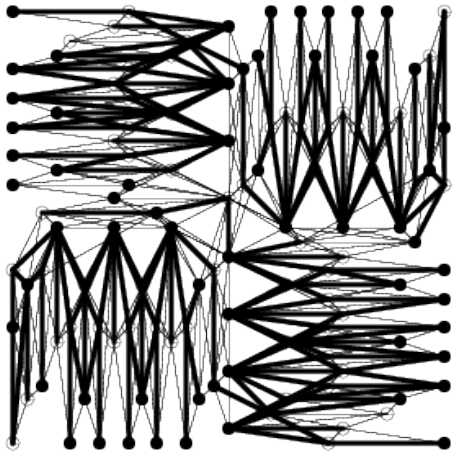

FIG. 3. *Graph on level four for Experiment 5.3* ($h = 1/32$).

5.6 The Adaptive Test Vector

For the numerical experiments, we used only the test vector $t = (1, \ldots, 1)^T$. Actually, the MLILU decomposition was developed for the application of different test vectors. In this section, we consider a simple example for the construction of more sophisticated test vectors. For a detailed description of the background and the idea behind this adaptive test vector scheme, we refer to [25].

As an example case, we consider the problem which yields the worst convergence of all experiments in the paper for the MLILU decomposition (see Experiment 5.6). Twelve iteration steps with the MLILU decomposition with respect to the test vector $t = (1, \ldots, 1)^T$ are computed. After that, we construct a new MLILU decomposition with respect to a new test vector t_{ad} which is the correction we added in the last iteration step to the approximate solution

$$t_{\text{ad}} = u^{(i)} - u^{(i-1)}$$

(see (1.1)). Then, one iteration step with the new MLILU decomposition is calculated (Algorithm 2.5 without smoothing steps, $\gamma = 1$). Finally, iteration steps with the initial MLILU decomposition are executed until the desired accuracy is reached.

EXPERIMENT 5.9. *We consider the conductivity distribution in Experiment 5.6 which yields the worst convergence rates for the MLILU decomposition* ($\gamma = 1$, $\sigma = 1/4$, $\lambda = 0.5$, $\sigma^2_{\log k_f} = 0.8$). *The convergence behavior is presented in Figure 5. The results for the adaptive test vector scheme are indicated by the dashed line.*

In Experiment 5.9, the adaptive test vector scheme needs 9 iterations less than the standard MLILU method. Since we have to construct and to store a second decomposition, the adaptive test vector scheme does not pay off for this example. However, we

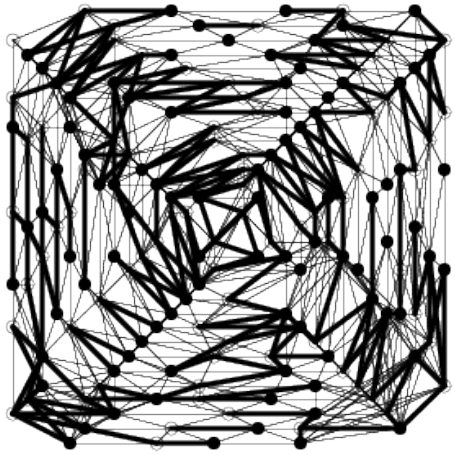

FIG. 4. *Graph on level four for Experiment 5.8 ($h = 1/32$).*

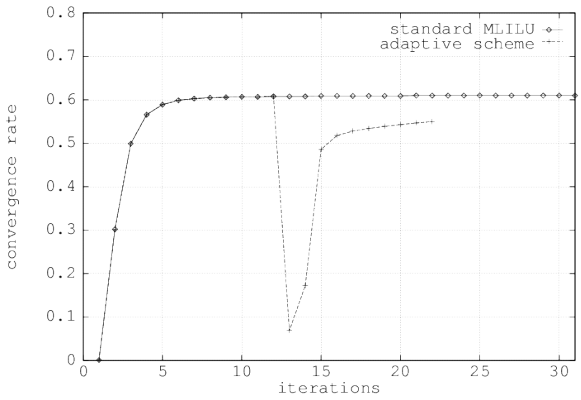

FIG. 5. *Convergence behavior for Experiment 5.9.*

believe that a more sophisticated choice of the test vector might improve the MLILU decomposition for more complex problems like 3D-problems and systems of differential equations. The MLILU decomposition for those problems is still under investigation.

References

[1] R. E. ALCOUFFE, A. BRANDT, J. E. DENDY AND J. W. PAINTER, *The multigrid-method for the diffusion equation with strongly discontinuous coefficients,* SIAM J. Sci. Stat. Comput. 2 (4), pp. 430–454, 1981.

[2] O. AXELSON AND P. VASSILEVSKI, *Algebraic multilevel preconditioning methods.* Part I: Numer. Math. 56, pp. 157–177, 1989; Part II: SIAM J. Numer. Anal. 27, pp. 1569–1590, 1990.

[3] O. AXELSON AND B. POLMAN, *Proceedings of the Conference on Algebraic Multilevel Iteration Methods with Applications,* University of Nijmegan, Nijmegan, The Netherlands, 1996.

[4] R. E. Bank and S. Gutsch, *Hierarchical basis for the convection-diffusion equation on unstructured meshes,* in Ninth International Symposium on Domain Decomposition Methods for Partial Differential Equations (P. Bjorstad, M. Espendal and D. Keyes eds.), J. Wiley and Sons, New York, to appear.

[5] R. E. Bank and J. Xu, *The hierarchical basis multigrid method and incomplete LU decomposition*, in Seventh International Symposium on Domain Decomposition Methods for Partial Differential Equations (D. Keyes and J. Xu, eds.), pp. 163–173, AMS, Providence, Rhode Island, 1994.

[6] R. E. Bank and R. K. Smith, *The hierarchical basis multigraph algorithm,* submitted to SIAM J. on Scientific Computing.

[7] P. Bastian, *Parallele adaptive Mehrgitterverfahren,* PhD. thesis, ICA-Bericht 94/1, Universität Stuttgart, 1994.

[8] P. Bastian, K. Birken, K. Johannsen, S. Lang, K. Eckstein, N. Neuss, H. Rentz-Reichert, C. Wieners, *UG - A flexible software toolbox for solving partial differential equations,* Computing and Visualization in Science 1, 1997.

[9] J. E. Dendy, *Black box multi-grid,* J. Comput. Physics, 48, pp. 366–386, 1982.

[10] I. Gustafsson, *A class of first order factorization methods,* BIT 18, pp. 142–156, 1978.

[11] W. Hackbusch, *Multi-grid methods and applications,* Springer-Verlag, Berlin, 1985.

[12] W. Hackbusch, *On first and second order box schemes,* Computing 41, pp. 277–296, 1989.

[13] W. Hackbusch, *Iterative solution of large sparse systems,* Springer, New York, 1994.

[14] P. W. Hemker and P. Wesseling, eds., *Multigrid methods, Proceedings of the fourth European Multigrid Conference,* INSM, Birkhäuser, Basel, 1994.

[15] A. Reusken, *Multigrid with matrix-dependent transfer operators for convection-diffusion problems,* in [14], 1994.

[16] A. Reusken, *Fourier analysis of a robust multigrid method for convection-diffusion equations*, Numerische Mathematik 71, pp. 365–398, 1995.

[17] A. Reusken, *A multigrid method based on incomplete Gaussian elimination,* J. Num. Lin. Alg. with Appl. 3, pp. 369–390, 1996.

[18] M. J. L. Robin, A. L. Gutjahr, E. A. Sudicky and J. L. Wilson, *Cross-correlated random field generation with the direct Fourier transform method,* Water Resources Research, Vol. 29, No. 7, pp. 2385–2397, 1993.

[19] J. W. Ruge and K. Stüben, *Algebraic multigrid,* in Multigrid methods (S. F. McCormick, ed.), SIAM, Philadelphia, 1987.

[20] A. van der Ploeg, E. Botta and F. Wubs, *Nested grids ILU-decomposition (NGILU),* J. Comput. Appl. Math. 66, pp. 515–526, 1996.

[21] C. Schwarz, *Effective parameter für adsoptiven Transport im Grundwasser,* Diploma thesis Universität Heidelberg, 1995.

[22] C. Wagner, W. Kinzelbach and G.Wittum, *Schur-complement multgrid — a robust method for groundwater flow and transport problems,* Numerische Mathematik 75, pp. 523–545, 1997.

[23] C. Wagner, *Frequenzfilternde Zerlegungen für unsymmetrische Matrizen und Matrizen mit stark variierenden Koeffizienten,* Ph. D. thesis Universität Stuttgart, ICA-Bericht 95/7, Stuttgart, 1995.

[24] C. Wagner, *Tangential frequency filtering decompositions for symmetric matrices*, Numerische Mathematik 78, pp. 119–142, 1997.

[25] C. Wagner and G. Wittum, *Adaptive filtering,* Numerische Mathematik 78, pp. 305-328, 1997.

[26] G. Wittum, *On the robustness of ILU smoothing,* SIAM J. Sci. Stat. Comp. 10, pp. 699–717, 1989.

[27] P. M. de Zeeuw, *Matrix-dependent prolongations and restrictions in a black box multigrid solver,* J. Comput. Appl. Math. 33, pp. 1–27, 1990.