RANA 96-20
October 1996

An approximate cyclic reduction multilevel preconditioner for general sparse matrices

by

A.A. Reusken
An approximate cyclic reduction multilevel preconditioner
for general sparse matrices

by

Arnold Reusken

Abstract. We discuss an iterative method for solving large sparse systems of equations. A hybrid method is introduced which uses ideas both from ILU preconditioning and from multigrid. The resulting preconditioning technique requires the matrix only. A multilevel structure is obtained by using maximal independent sets for graph coarsening. For Schur complement approximation on coarser graphs an incomplete Gaussian elimination is used. The resulting preconditioner has a transparent modular structure similar to the algorithmic structure of a multigrid V-cycle.

Key words. algebraic multigrid, cyclic reduction, incomplete Gaussian elimination, independent set orderings.

AMS subject classification. 65F10, 65N20.

1. Introduction

Multigrid methods are very efficient iterative solvers for the large systems of equations resulting from discretizing partial differential equations (cf. [14], [27] and the references therein). An important principle of multigrid is that a basic iterative method which yields appropriate local corrections is applied on a hierarchy of discretizations with different characteristic mesh sizes. This multilevel structure is of main importance for the efficiency of multigrid. Another class of efficient iterative solvers consists of Krylov subspace methods combined with ILU preconditioning (cf. [7] and the references therein). These methods only need the matrix and are in general easier to implement than multigrid methods. Also the Krylov subspace methods are better suitable for a "black-box" approach. On the other hand, for discretized partial differential equations the Krylov methods with ILU preconditioning are often less efficient than multigrid methods.

In the multigrid field there have been developed methods which have a multilevel structure but require only the matrix of the linear system. These are called algebraic multigrid methods. Approaches towards algebraic multigrid are presented in [6], [12], [22], [26]. In all these methods one tries to mimic the multigrid principle. First one introduces a "reasonable" coarse "grid" space. Then a prolongation operator is chosen and for the restriction one usually takes the adjoint of the prolongation. The operator on the coarse grid space is defined by a Galerkin approach. With these components, a standard multigrid approach (smoothing + coarse grid correction) is applied. These algebraic multigrid methods can be used in situations where a grid (hierarchy) is not available. Also these methods can be used for developing black-box solvers.
Recently there have been developed ILU type of preconditioners with a multilevel structure, cf. [5], [18], [23]. The multilevel structure is induced by a level wise numbering of the unknowns.

In this paper we present a new hybrid method which uses ideas both from multilevel ILU preconditioning and from multigrid. We use level wise ordering as in [5], [18], [23]. For constructing equations on coarser levels we apply a technique which is used in the multigrid methods in [19], [20], [21]. For the presentation and implementation of the method we use the framework of a multigrid V-cycle. Our method is algebraic, i.e. only the matrix is required.

Our starting point is the classical cyclic reduction (CR) method for solving a system with a tridiagonal matrix. The algorithmic structure of this CR method is very similar to the structure of a multigrid V-cycle algorithm. Note, however, that the basic ideas underlying these methods are very different. Generalization of the CR method to a method applicable to general sparse matrices is based on two ideas. Firstly, for obtaining a red-black ("fine-coarse") partitioning of the vertices in the graph we use a maximal independent set approach. This type of graph coarsening method has recently been successfully applied in sparse matrix solvers, cf. [8], [9], [11], [23]. Secondly, for approximating the Schur complement on the set of black vertices (which results from the graph coarsening algorithm) we apply an incomplete Gaussian elimination method. This method is similar to the one used in [19], [20], [21]. In these papers, however, a grid hierarchy is assumed, whereas in the present paper we require the matrix only.

As in multigrid we use approximate inner solvers (smoother in multigrid) for the systems of equations in the red vertices. An important property of the resulting preconditioner is a transparent modular structure.

We give a detailed description of the approximate cyclic reduction preconditioner. We discuss properties related to sparsity, stability and approximation quality. Due to the modular structure there are many obvious variants (modifications) of the preconditioner. In this paper we give numerical results only for a basic form of the preconditioner. Testing and analyzing modifications of this preconditioner is a subject of current research.

2. Cyclic reduction

We recall the classical method of cyclic reduction. This method can be used, for example, for solving a linear system with a tridiagonal matrix or with a special block tridiagonal matrix (cf. [13], [16], [24], [25]). We explain the cyclic reduction principle by considering an $n \times n$ linear system with a tridiagonal matrix:
Reordering the unknowns based on an obvious red-black (or "odd-even") structure results in a permuted system with a matrix of the form

\[
(2.2) \quad \begin{pmatrix}
D_b & B_{br} \\
B_{rb} & D_r
\end{pmatrix},
\]

in which \([D_b, B_{br}]\) represents the equations in the unknowns with a black label and \([B_{rb}, D_r]\) represents the equations in the unknowns with a red label. Note that \(D_b, D_r\) are diagonal matrices. Gaussian elimination in the red points results in a reduced system with dimension (approximately) \(\frac{1}{2}n\). In matrix notation this corresponds to block UL-decomposition:

\[
(2.3) \quad \begin{pmatrix}
S_b & 0 \\
0 & I
\end{pmatrix}
\begin{pmatrix}
B_{rb} & D_r
\end{pmatrix}, \quad S_b := D_b - B_{br}D_r^{-1}B_{rb}.
\]

The reduced system has a matrix \(S_b\) (Schur complement) which is tridiagonal, and thus the same approach can be applied to \(S_b\). So the basic cyclic reduction idea is to reduce significantly the dimension of the problem repeatedly until one has a relatively small problem that can be solved easily. This small system is then solved and the previously eliminated (red) unknowns are found by a simple back-substitution process. Note that cyclic reduction is equivalent to Gaussian elimination applied to a permuted system of equations and that different implementations are possible (cf. [13], [25]).

When solving a system as in (2.1) with cyclic reduction, one usually adapts the right hand side in the reduction phase. For example, in the first reduction step the original system is transformed to

\[
(2.4) \quad \begin{pmatrix}
S_b & 0 \\
B_{rb} & D_r
\end{pmatrix} \begin{pmatrix}
P_x
\end{pmatrix} = \begin{pmatrix}
I & -B_{br}D_r^{-1}
\end{pmatrix} \begin{pmatrix}
P_b
\end{pmatrix}.
\]

In such a situation we do not need to store the coefficients of \(B_{br}\). In our approach, however, we will need both the upper and the lower triangular part of the UL-decomposition (as in ILU preconditioners). Thus we consider a cyclic reduction algorithm in which the block UL-decomposition as in (2.3) is computed. This UL-decomposition is then used to solve the system with a backward-forward elimination process.
In the remainder of this section we give a quite detailed description of an algorithm based on cyclic reduction. This is not meant to be a very efficient implementation of the cyclic reduction method. We present this algorithm because it makes the (implementation of) the approximate cyclic reduction preconditioner, discussed in Section 4, much easier to understand.

The algorithm consists of two phases: a decomposition phase and a solution phase. In the decomposition phase we only need the tridiagonal matrix $A \in \mathbb{R}^{n \times n}$. In the solution phase we need the right hand side $b$ and the decomposition resulting from the decomposition phase.

**Decomposition phase.** We assume $A \in \mathbb{R}^{n \times n}$ as in (2.1). To $A$ there corresponds an ordered graph with vertex set denoted by $\{v_1, v_2, ..., v_n\}$. $\text{Bound}$, with $1 < \text{Bound} < n$ is a given integer (used in D5 below). Set $i := 1$, $S_1 := A$, $m_0 := n$.

**D1. Red-black partitioning of the vertex set.** Given the graph of $S_i$ we make a red-black partitioning of the vertices. This results in $n_i$ vertices with label red, and $m_i$ vertices with label black. Note: $m_i + n_i = m_{i-1}$.

**D2. Determine permutation.** We determine a symmetric permutation $p_i : \{1, 2, ..., m_{i-1}\} \rightarrow \{1, 2, ..., m_{i-1}\}$ such that applying this permutation to the index set of the vertices results in an ordering in which all vertices with label red have index $j \in (m_i, m_{i-1}]$ and all vertices with label black have index $j \in [1, m_i]$. Note that since we only have to permute between the sets $\{j \mid j > m_i \text{ and } \text{label}(v_j) = \text{black}\}$ and $\{j \mid j \leq m_i \text{ and } \text{label}(v_j) = \text{red}\}$, such a permutation can be fully characterized by a permutation $\tilde{p}_i : \{m_i + 1, m_i + 2, ..., m_{i-1}\} \rightarrow \{1, 2, ..., m_i\}$.

**D3. Determine permuted matrix.** The symmetric matrix corresponding to the permutation $p_i$ of D2 is denoted by $P_i$. We determine $P_i S_i P_i$. This matrix has a $2 \times 2$-block representation:

\[
P_i S_i P_i = \begin{bmatrix} B_i & D_i \\
C_i & A_i \end{bmatrix},
\]

with $A_i \in \mathbb{R}^{m_i \times m_i}$, $B_i \in \mathbb{R}^{m_i \times m_i}$, $C_i \in \mathbb{R}^{m_i \times m_i}$, $D_i \in \mathbb{R}^{m_i \times m_i}$.

**D4. Compute Schur complement.** Compute the Schur complement $S_{i+1} \in \mathbb{R}^{m_i \times m_i}$ in the vertices with label black:

\[S_{i+1} = B_i - D_i A_i^{-1} C_i.\]

**D5. Store.** Save $m_i, \tilde{p}_i, A_i, C_i, D_i$. If $m_i < \text{Bound}$ then save $S_{i+1}$ (stop the reduction process) else $i := i + 1$ and goto D1.

If this decomposition process stops with $i = i_{\text{max}}$, we obtain integers $m_1 > m_2 > ... > m_{i_{\text{max}}}$, permutation vectors $\tilde{p}_i$ ($1 \leq i \leq i_{\text{max}}$), sparse matrices $A_i, C_i, D_i$ ($1 \leq i \leq i_{\text{max}}$) and the Schur complement matrix on the highest level $S_{i_{\text{max}}+1}$. We use the following
terminology: \( \hat{p}_i \) is called the permutation operator on level \( i \), \( A_i \) is called the solve operator on level \( i \), \( C_i \) is called the collect operator on level \( i \), \( D_i \) is called the distribute operator on level \( i \).

The red vertices on all levels, together with the black vertices on the final level induce a direct sum decomposition \( \mathbb{R}^n = \mathbb{R}^{n_1} \oplus \mathbb{R}^{n_2} \oplus \ldots \oplus \mathbb{R}^{n_{i_{\text{max}}}} \oplus \mathbb{R}^{n_{i_{\text{max}}}} \). The vertices on level \( i \) with label red are assigned the level number \( i \), and the vertices on level \( i_{\text{max}} \) with label black are assigned level number \( i_{\text{max}} + 1 \). The vertices (unknowns) with level number \( j \) are called the level \( j \) vertices (unknowns). Note that every vertex has a unique level number.

**Remark 2.1.** In the decomposition phase there are arithmetic operations only in the computation of the Schur complement (D4). In the other steps we perform only labeling and renumbering. There is a natural relation between the information stored in step D5 and the set of vertices of the graph corresponding to \( A \). Consider an unknown (vertex) with level number \( \ell \). If \( \ell = i_{\text{max}} + 1 \) there is a natural relation between this unknown and a row of \( S_{i_{\text{max}}+1} \). If \( \ell \leq i_{\text{max}} \) there is a natural relation between this unknown and one element of \( \hat{p}_i \), a row of \( A_i \), a row of \( C_i \) and a column of \( D_i \). So all the information of \( \hat{p}_i, A_i, C_i, D_i \) \((1 \leq i \leq i_{\text{max}})\) can be stored by assigning to a level \( i \) vertex one element of \( \hat{p}_i \), one row of \( A_i \), one row of \( C_i \) and one column of \( D_i \). The amount of information thus stored at a level \( i \) vertex is fully determined by the sparsity of the Schur complement \( S_i \) on level \( i \).

**Solution phase.** For a clear description of the solution phase we introduce permute, collect, distribute and solve operations. These operations use the corresponding operators which are available from the decomposition phase. We give a description in a pseudo-programming language.

```plaintext
procedure permuteoperation(i: integer; var x ∈ \mathbb{R}^{m_{i-1}}) (* uses \( \hat{p}_i \))
  for j := m_i + 1 to m_{i-1} do
    if j ≠ \( \hat{p}_i(j) \) then interchange \( x_j \) and \( x_{\hat{p}_i(j)} \);

procedure collectoperation(i: integer; var x ∈ \mathbb{R}^{m_i}; g ∈ \mathbb{R}^{m_i}) (* uses \( C_i \))
  compute x := x - C_i g;

procedure distributeoperation(i: integer; var x ∈ \mathbb{R}^{m_i}; g ∈ \mathbb{R}^{n_i}) (* uses \( D_i \))
  compute x := x - D_i g;

procedure solveoperation(i: integer; var x ∈ \mathbb{R}^{n_i}) (* uses \( A_i \))
  solve \( A_i w = x \); x := w;

procedure highestlevelsolve(var x ∈ \mathbb{R}^{n_{i_{\text{max}}}}) (* uses \( S_{i_{\text{max}}+1} \))
  solve \( S_{i_{\text{max}}+1} w = x \); x := w;
```

Using these procedures it is easy to formulate the backward and forward substitution process, i.e. the solution phase, of the cyclic reduction method. On each level \( i \) \((1 \leq i \leq i_{\text{max}} + 1)\) we define ULsolve as follows:
procedure ULsolve(i: integer; var f ∈ ℝ^n_i);
(* solves ULx = f; f := x*)
var f_red ∈ ℝ^n_i;
begin
  if i = i_max + 1 then highestlevelsolve(f) else
  begin
    permuteoperation(i, f);
    partition f = (f_b, f_r) with f_r ∈ ℝ^n_i, f_b ∈ ℝ^n_b;
    make a copy f_red := f_r;
    solveoperation(i, f_red);
    distributeoperation(i, f_b, f_red);
    ULsolve(i + 1, f_b);
    collectoperation(i, f_r, f_b);
    solveoperation(i, f_r);
    permuteoperation(i, f);
  end
end;

For solving Ax = b we use ULsolve(1, b). The structure of ULsolve is similar to the structure of the multigrid V-cycle algorithm as presented in [14]. The distribute and collect operations correspond to the multigrid restriction and prolongation respectively. The solve operation corresponds to the smoother in multigrid. Note, however, that in ULsolve every unknown is involved in the solve operations of precisely one level (as in hierarchical basis multigrid, cf. [2]). The implementation of cyclic reduction presented in this section clearly has a modular structure. In the next section we introduce modifications of a few components of this cyclic reduction algorithm. These modifications then lead to the approximate cyclic reduction preconditioner of Section 4.

3. Modifications of cyclic reduction

In Section 2 we discussed a cyclic reduction algorithm for solving a system with a tridiagonal matrix. In this section we assume S to be a general sparse matrix, and we consider the cyclic reduction algorithm of Section 2. As a typical example for S one can think of a discretized partial differential operator. We assume diag(S) to be nonsingular.

Below, in Section 3.1, we discuss a known graph coarsening method which can be used for the construction of a red-black type of structure. In Section 3.2 we present a method for approximating the Schur complement in the black vertices. This approximation has properties similar to the original matrix. In particular, sparsity is preserved. In Section 3.3 we consider the approximate solution of the systems in the red vertices (cf. solveoperation in §2).
3.1. Graph coarsening based on independent set orderings. Graph theory provides some simple tools for creating a structure which is similar to the coarse-fine grid structure used in multigrid. One such a tool which recently has been applied in sparse solution techniques (cf. [5], [11], [23]) is based on so called maximal independent sets. We briefly explain this notion (cf. [10]). A graph \( G = G(V, E) \) consists of a finite set \( V \) of vertices together with a set \( E \) of unordered pairs of vertices called edges. If \( e = \{u, v\} \) is an edge of \( G \), the \( u \) and \( v \) are adjacent vertices. Two vertices that are not adjacent are said to be independent. A set \( T \) of vertices is independent if every two vertices of \( T \) are independent. \( T \) is called a maximal independent set of vertices if \( T \) is independent but no proper superset of \( T \) is independent. An equivalent definition is: \( T \) is a maximal independent set (of vertices of \( G(V, E) \)) if and only if no vertex in \( T \) is adjacent to any other vertex in \( T \), while every vertex in \( V \setminus T \) is adjacent to at least one vertex in \( T \). Note that a maximal independent set is in general not unique.

A directed graph or digraph \( D = D(V, E) \) consists of a finite set \( V \) of vertices together with a set \( E \) of ordered pairs of vertices called arcs (or directed edges). If \( a = (u, v) \) is an arc of \( D \), \( u \) is said to be adjacent to \( v \) and \( v \) is said to be adjacent from \( u \). The underlying graph of a digraph \( D \) is that graph obtained by replacing each arc \((u, v)\) or symmetric pair \((u, v), (v, u)\) of arcs by an edge \( e = \{u, v\} \). A set of vertices \( T \) of a digraph \( D \) is called a maximal independent set if and only if \( T \) is a maximal independent set of the underlying graph of \( D \).

Different techniques for constructing a maximal independent set are known. Several possible algorithms are given in [11], [23]. In view of our applications we consider a directed graph \( D(V, E) \). We sketch our method for constructing a maximal independent set \( T \). The algorithm consists of a graph traversal, i.e. visiting all the vertices of \( D \) in a systematic way, and a labeling method. One well-known algorithm for graph traversal is the breadth first search (BFS), cf. [17]. The BFS algorithm starts with a vertex \( v \) and marks it as visited. Then unvisited vertices adjacent from \( v \) are visited next. Then unvisited vertices adjacent from these vertices are visited and so on. This approach is applied to every connected component of \( D \). A detailed description can be found in [17]. In the following we explain how labeling is done. We initialize with label(\( v \)) := white for all \( v \in V \). Let \( v_0 \) be the currently visited vertex in the BFS algorithm. If \( v_0 \) is an isolated vertex, i.e. there are no vertices \( w \neq v_0 \) adjacent from \( v_0 \), we define label(\( v_0 \)) := red. Otherwise:

```plaintext
if label(\( v_0 \)) = white then
    if label(\( w \)) \in \{white, red\} for all \( w \) adjacent from \( v_0 \) then
        label(\( v_0 \)) := black;
        label(\( w \)) := red for all \( w \) adjacent from \( v_0 \)
    endif else
        label(\( v_0 \)) := red
    endif.
```

This results in a red-black partitioning of the vertex set \( V \) and the set of black vertices is a maximal independent set.

In our approach the black vertices are used as vertices of a coarse graph. This is an
important difference with the approach in [23], where the red vertices are used for the coarse graph. Note that if we have an underlying two-level grid structure, then often the vertices corresponding to the coarse grid points form a maximal independent set.

3.2. Schur complement approximation based on incomplete Gaussian elimination. In [19],[20],[21] one can find a technique for constructing a sparse approximation of a coarse grid Schur complement. In these papers a grid structure is assumed and the Schur complement approximation is used in a special multigrid solver. In this section we show that the same technique can be used for approximating the Schur complement in the case of a matrix graph with a two-level structure as resulting from the maximal independent set partitioning. A similar ILU based coarsening technique, which however uses grid information, is presented in [3].

We first explain how we approximate the Schur complement if we have a grid structure (cf. [21]). We consider a setting in which we have a "coarse" mesh $\Omega_H$ and a "fine" mesh $\Omega_h$ that is obtained after a standard refinement of $\Omega_H$. Furthermore we assume a given stiffness matrix $S_h$ which results from a discretization method on $\Omega_h$ with linear finite elements on triangles or with nine-point finite differences on a square grid. We now consider Gaussian elimination from a graph theoretical point of view. To obtain the Schur complement system we should eliminate all arcs $(C, F)$, with $C$ a vertex corresponding to a grid point in $\Omega_h \setminus \Omega_H$ and $F$ a vertex corresponding to a grid point in $\Omega_h \setminus \Omega_H$ (note that we have a directed graph).

We consider an arbitrary arc $(C, F)$ (cf. Figure 1a) and we perform an incomplete Gaussian elimination as follows. First we apply a standard Gaussian elimination step using the equation at $F$ to eliminate the arc $(C, F)$. This results in fill-in arcs as shown in Figure 1b. Now fill-in arcs $(C, G)$ with $G$ a vertex corresponding to a grid point in $\Omega_h \setminus \Omega_H$ are eliminated using a simple linear interpolation process. For example, an arc $(C, G)$ as in Figure 1b with associated value $\alpha$ is removed and replaced by new arcs $(C, D_1), (C, D_2)$ with associated value $\frac{1}{2}\alpha$. An arc $(C, G_1)$ (cf. Figure 1b) with associated value $\alpha$ is replaced by arcs $(C, D_1), (C, D_2), (C, D_3), (C, C)$ with associated value $\frac{1}{2}\alpha$. So after this incomplete elimination step we have removed the arc $(C, F)$ and created only fill-in arcs between vertices corresponding to coarse grid points (cf. Figure 1c). Applying this elimination process for all arcs $(C, F)$ results in a decoupling of the coarse grid unknowns from the unknowns corresponding to grid points in $\Omega_h \setminus \Omega_H$. For the Schur complement approximation $\tilde{S}_H$ we use the resulting system of coarse grid equations.

Remark 3.1. Another description of the Schur complement approximation is as follows.
After reordering, the stiffness matrix can be represented in 2 x 2-block form as (cf. (2.5))

\[
\begin{bmatrix}
    B & D \\
    C & A
\end{bmatrix}
\]

The coarse grid Schur complement system is given by \( \mathbf{S}_H = \mathbf{B} - \mathbf{D} \mathbf{A}^{-1} \mathbf{C} \). Let \( \mathbf{A}_d := \text{diag}(\mathbf{A}) \) and define \( \mathbf{J}_{hH} : \ell^2(\Omega_H) \to \ell^2(\Omega_h \setminus \Omega_H) \) to be the interpolation operator which determines the value at a vertex in \( \Omega_h \setminus \Omega_H \) using (bi)linear interpolation of the values at the parents in \( \Omega_H \). The exact Schur complement \( \mathbf{S}_H \) is (by definition) the result of block Gaussian elimination:

\[
\begin{bmatrix}
    \mathbf{I} & -\mathbf{D} \mathbf{A}^{-1} \\
    \emptyset & \mathbf{I}
\end{bmatrix}
\begin{bmatrix}
    \mathbf{S}_H & \emptyset \\
    \mathbf{C} & \mathbf{A}
\end{bmatrix} = \mathbf{P}_h \mathbf{S}_h \mathbf{P}.
\]

Applying point Gaussian elimination at the vertices of \( \Omega_h \setminus \Omega_H \) (i.e. the red points) corresponds to replacing \( \mathbf{A}^{-1} \) by \( \mathbf{A}_d^{-1} \). So point Gaussian elimination yields, with \( \Delta := \mathbf{I} - \mathbf{A}_d^{-1} \mathbf{A} \):

\[
\begin{bmatrix}
    \mathbf{I} & -\mathbf{D} \mathbf{A}_d^{-1} \\
    \emptyset & \mathbf{I}
\end{bmatrix}
\begin{bmatrix}
    \mathbf{B} & \mathbf{D} \Delta \\
    \mathbf{C} & \mathbf{A}
\end{bmatrix} =: \hat{\mathbf{S}}_h.
\]

Note that the Schur complement of the transformed matrix \( \hat{\mathbf{S}}_h \) is equal to \( \mathbf{S}_H \). We now use lumping of the fill-in block \( \mathbf{D} \Delta \) to the (1,1) diagonal block, i.e. \( \hat{\mathbf{S}}_h \) is approximated by
This $\hat{S}_h$ (which is the Schur complement of $S_h$) is precisely the Schur complement approximation described above. Yet another description of this Schur complement approximation is given in [21], Remark 3.2.

The method for Schur complement approximation is just point Gaussian elimination (in the red vertices) with lumping of certain fill-in to "parents". The grid structure is used only for the choice of reasonable parents. It is straightforward to apply a similar approach to a matrix graph with a two-level red-black structure, in which the black vertices play the role of the coarse grid points. For lumping of fill-in we want every red vertex to have at least one "reasonable" black parent vertex. In the red-black structure obtained with the maximal independent set approach, every red vertex is adjacent to or from at least one black vertex. So at least one candidate parent is available.

Consider an arbitrary red vertex $v_r$. We introduce the neighborhood of black vertices adjacent from $v_r$:

$$N_{\text{black}}(v_r) := \{ v \in V \mid \text{label}(v) = \text{black and } v \text{ adjacent from } v_r \} .$$

The situation that for $v_r$ we have $N_{\text{black}}(v_r) = \{ \emptyset \}$ occurs only seldom. For such a vertex we use a special treatment. In our implementation, fill-in to such a vertex is removed by setting the matrix entries to zero (so we do not use lumping). If $|N_{\text{black}}(v_r)| = 1$, then the element of $N_{\text{black}}(v_r)$ is called the parent of $v_r$ and fill-in is lumped to this parent. If $|N_{\text{black}}(v_r)| = 2$, then the two elements of $N_{\text{black}}(v_r)$ are taken as parents of $v_r$ and fill-in corresponding to arcs to $v_r$ is distributed equally (i.e. with weight $\frac{1}{2}$) between these two parents. If $|N_{\text{black}}(v_r)| > 2$ we select two parents by considering the size of the matrix entries: $|s_{v_r,w}|, w \in N_{\text{black}}(v_r)$. The two largest values determine two parents of $v_r$.

The approximate Schur complement is obtained by combining point Gaussian elimination with this lumping method. We give a representation of this approximate Schur complement as in Remark 3.1. The lumping method yields a "restriction" $R := R_{V_r}^{V_r} \to R_{V_b}^{V_b}$, with

$$V_r := \{ v \in V \mid \text{label}(v) = \text{red} \} , \text{ and } V_b := \{ v \in V \mid \text{label}(v) = \text{black} \} .$$

We define $J_{rb} = R^T$. The given matrix $S$ (with digraph $D(V, E)$) has a block $2 \times 2$ representation corresponding to the red-black partitioning of $V$:

(3.1) $\text{PSP} = \begin{bmatrix} B & D \\ C & A \end{bmatrix}$.
We define \( A_d := \text{diag}(A), S_b := B - DA_d^{-1}C \). The same derivation as in Remark 3.1 can be applied and we obtain the following representation for the approximate Schur complement \( \tilde{S}_b \):

\[
(3.2) \quad \tilde{S}_b = B - DA_d^{-1}C + D(I - A_d^{-1}A)J_{rb} \, .
\]

We now discuss some properties of \( \tilde{S}_b \) related to sparsity, stability and approximation quality. Clearly the construction of \( \tilde{S}_b \) aims at a sparse approximation of \( S_b \). If we consider pde's and we assume an underlying standard multilevel grid structure, it is easy to verify that at all coarser grids \( \tilde{S}_b \) has approximately the same sparsity as the matrix on the finest grid (cf. [19], [21]). In case of a "general" matrix graph it is not possible to give an a priori realistic indication of the sparsity pattern of \( \tilde{S}_b \). Results of numerical experiments are given in Section 5.

A result related to stability of \( \tilde{S}_b \) is given in Lemma 3.2.

**Lemma 3.2.** Assume that for \( S = (s_{ij}) \) as in (3.1) we have:

\[
(3.3a) \quad s_{ij} \leq 0 \quad \text{for all} \quad i \neq j ,
\]

\[
(3.3b) \quad \sum_{j \neq i} |s_{ij}| \leq s_{ii} \quad \text{for all} \quad i .
\]

Then for \( \tilde{S}_b = (\tilde{s}_{ij}) \) as in (3.2) the properties (3.3) hold with \( s_{ij} \) replaced by \( \tilde{s}_{ij} \).

**Proof.** In the proof we use componentwise inequalities for matrices and vectors. We use the notation: \( e_r = (1,1,...,1)^T \) with length the number of red vertices, and \( e_b = (1,1,...,1)^T \) with length the number of black vertices. From (3.3a) we obtain: \( D \leq 0, C \leq 0 \). Using (3.3b) we get \( A_d \geq 0 \) and thus \( DA_d^{-1}C \geq 0 \), \( I - A_d^{-1}A \geq 0 \). From the definition of \( J_{rb} \) we see \( J_{rb} \geq 0 \). Now it follows that

\[
DA_d^{-1}C - D(I - A_d^{-1}A)J_{rb} \geq 0
\]

holds. Combining this with \( (B)_{ij} \leq 0 \) for all \( i \neq j \) we obtain \( (\tilde{S}_b)_{ij} \leq 0 \) for all \( i \neq j \). From the definition of \( J_{rb} \) we deduce

\[
0 \leq J_{rb}e_b \leq e_r .
\]

Using this we obtain:

\[
A_d e_r + (C - (A_d - A)J_{rb})e_b \geq A_d e_r + Ce_b - (A_d - A)e_r = Ae_r + Ce_b \geq 0 .
\]

This yields

\[
(A_d^{-1}C - (I - A_d^{-1}A)J_{rb})e_b \geq -e_r .
\]
Thus we get
\[ \tilde{S}_b e_b = B e_b - D (A_d^{-1}C - (I - A_d^{-1}A) J_{rb}) e_b \geq B e_b + D e_r \geq 0. \]

So (3.3b) holds with \( s_{ij} \) replaced by \( (\tilde{S}_b)_{ij} \). \( \square \)

**Remark 3.3.** Probably it will be hard to develop a satisfactory theoretic analysis concerning the quality of \( \tilde{S}_b \) as a preconditioner for \( S_b \). In [19], [21] we analyzed \( S_b^{-1} S_b \) in a model setting where Fourier analysis is applicable. We summarize the main result of [21]. On \((0,1)^2\) we assume a uniform square coarse mesh \( \Omega_H \) with mesh size \( H \) ("black vertices") and a refined mesh \( \Omega_h \) with mesh size \( h = \frac{1}{2} H \) ("red vertices": \( \Omega_h \setminus \Omega_H \)). We consider a periodic problem with a constant 5-point stable difference star:

\[ (3.4) \begin{bmatrix}
-\alpha_1 & 0 & \cdots & 0 \\
0 & -\alpha_2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & -\alpha_n
\end{bmatrix}, \quad \alpha_i > 0, \quad \sum_i \alpha_i = 1. \]

We use standard coarsening \((h \to 2h)\) and thus the Schur complement on the coarse grid \( \tilde{S}_b \) will be a full matrix. If we apply the incomplete Gaussian elimination technique described above, we obtain the approximation \( \tilde{S}_b \) of \( S_b \). The approximate Schur complement \( \tilde{S}_b \) has a constant 9-point difference star. A (tedious) Fourier analysis shows that for the spectrum of \( S_b^{-1} \tilde{S}_b \) we have the result:

\[ \sigma(S_b^{-1} \tilde{S}_b) \subset \{ z \in C \mid |z - 1| \leq 1 \wedge \text{Re}(z) \geq \frac{1}{2} \}. \]

This yields a strong robustness result: for all constant 5-point difference stars as in (3.4) (e.g. diffusion, anisotropic diffusion, convection-diffusion) the Schur complement approximation \( \tilde{S}_b \) is an "optimal" preconditioner for \( S_b \). \( \square \)

### 3.3 Inner solver.

In the procedure **solveoperation** of the cyclic reduction algorithm we have to solve a system of equations in the red vertices, which we call the "red vertices system" here. On level 1 this system results by taking the original equations in the red vertices and deleting all coefficients in these equations related to arcs from the red vertices to black vertices. On higher levels the same holds but now with the (approximate) Schur complement equations instead of the original equations. Deleting these coefficients strongly enhances diagonal dominance and it appears to be reasonable to use a simple basic iterative method for solving these red vertices systems \emph{approximately}. For the case of a diffusion problem and assuming a linear finite element discretization hierarchy, it is proved in [1] that these red vertices systems have condition number \( O(1) \) \((h \downarrow 0)\).

Since our coarsening strategy is based on the graph structure only (not using the values of the nonzero matrix entries), the red vertices systems will be relatively hard to solve for problems with strong anisotropy or with strong convection along grid lines ("alignment"). This is the same phenomenon as encountered in multigrid methods when we
use coarsening (or refinement) based on geometric information only. In the multigrid field one usually applies so called robust smoothers to deal with this problem (cf. [14], [27]). In our setting we can apply similar techniques as proposed in the multigrid field. In our opinion, a very powerful approach is to use an appropriate (matrix based) vertex ordering combined with a Gauss-Seidel or ILU solver. Recently, new ordering techniques have been presented in [15] (cf. also [4]). In our preconditioner we have not yet implemented these ordering techniques. We simply use a Gauss-Seidel method based on the given vertex ordering.

A major problem, which we did not solve yet, is the accuracy with which the red vertices systems should be solved. In our current implementation we take a fixed number of Gauss-Seidel iterations. From numerical experiments we concluded that an iteration number between 4 and 8 seems to be reasonable. We note that by far most of the arithmetic work of the approximate cyclic reduction preconditioner is needed in the Gauss-Seidel iterations for solving the red vertices systems approximately.

4. Approximate cyclic reduction preconditioner

In this section we reconsider the cyclic reduction algorithm of Section 2 and use the modifications discussed in Section 3. This results in a cyclic reduction type of preconditioner with a much larger range of applicability than the (classical) cyclic reduction method of Section 2. For the presentation of the preconditioner we use the setting of Section 2.

**Decomposition phase.** Let \( S \in \mathbb{R}^{n \times n} \) be given.

D1. **Red-black partitioning of the vertex set.** We use a graph coarsening method. In our implementation we use the method based on a maximal independent set, explained in Section 3.1.

D2. **Determine permutation.** As in Section 2.

D3. **Determine permuted matrix.** As in Section 2.

D4. **Compute Schur complement approximation.** We compute a sparse approximation of the Schur complement based on the incomplete Gaussian elimination technique explained in Section 3.2.

D5. **Store.** As in Section 2.

**Solution phase.** We use the procedure `ULsolve` as in Section 2. All procedures used in `ULsolve` remain the same, except `highestlevelsolve` and `solveoperation`. In these two procedures we now use approximate solvers. The system on the highest level has low dimension and can be solved accurately with negligible computational costs. In the procedure `solveoperation` we use the approach explained in Section 3.3.
This modified cyclic reduction method, which in general solves a system $Sx = f$ only approximately, is used as a preconditioner for an outer iterative method (e.g. GMRES). Then the operations in the decomposition phase are performed only once, whereas the procedure ULsolve will be called repeatedly.

With respect to the implementation of this preconditioner we want to emphasize two important properties. Firstly, if one has implemented the classical (and simple) cyclic reduction method for a tridiagonal matrix along the lines as explained in Section 2, one almost immediately obtains an implementation of the approximate cyclic reduction preconditioner (for a general sparse matrix). Having the implementation of cyclic reduction as in Section 2, we only need to replace a few procedures (e.g. coarsening in D1) by modified procedures. The changes are local and do not depend on the data structure. Secondly, the approximate cyclic reduction preconditioner has a modular structure. Therefore it is easy to change certain components, for example the graph coarsening method or the method for solving the red vertices systems. The modular structure is as in most multigrid methods. Note, however, that in the setting here the different components (procedures) only use sparse matrices and graphs, whereas in multigrid certain components (e.g. prolongation) need grid information, too.

Concerning data storage we note that Remark 2.1 applies to the approximate cyclic preconditioner, too.

**Modifications.** We discuss a few possible modifications which might be of interest for future research.

As in multigrid methods for problems with strong anisotropics, one may consider to use a graph coarsening method which takes into account the values of the nonzero entries in the matrix (cf. semicoarsening in [14] or the technique in [22]). One simple idea in this direction is to set relatively small entries in the matrix to zero (in the coarsening method only) and then apply the method of Section 3.1.

In the incomplete Gaussian elimination method used for approximating the Schur complement, there are some obvious alternatives. In the current implementation we remove fill-in by dividing it equally among two parents. As in matrix-dependent prolongations used in multigrid, one can consider a weighted distribution of fill-in. Also, as is common practice in ILU preconditioners, one can introduce a drop tolerance for reducing the number of nonzero entries in the approximate Schur complement.

As already noted at the end of Section 3.3, efficiency can probably be enhanced significantly by using a more appropriate solver for the red vertices systems. One might consider, for example, a GMRES($m$) or BICGSTAB method preconditioned with ILU (based on a matrix dependent ordering).

5. Numerical experiments

In this section we show results of a few numerical experiments with the approximate cyclic reduction preconditioner of Section 4. We only use the basic form of the preconditioner, and we do not consider any modifications as discussed at the end of Section 4. A systematic study of such modified versions is a topic of current research.
Experiment 1. We consider:

\[
\begin{cases}
-\varepsilon \Delta u + a(x, y)u_x + b(x, y)u_y = f & \text{in } \Omega = (0,1)^2, \\
    u = 0 & \text{on } \partial \Omega.
\end{cases}
\]

The functions \(a, b\) are defined by: \(a(x, y) = 0.1\) if \((x, y) \in (0.5,0.8)^2\) and \(a(x, y) = 100\) otherwise; \(b(x, y) = 0.2\) if \((x, y) \in (0.5,0.8)^2\) and \(b(x, y) = 200\) otherwise.

We use a finite difference discretization on a uniform square mesh with mesh size \(h\). This results in a discrete operator with stencil:

\[
[S] = \frac{\varepsilon}{2h^2} \begin{bmatrix}
-\frac{1}{2} & -1 & -\frac{1}{2} \\
-1 & 6 & -1 \\
-\frac{1}{2} & -1 & -\frac{1}{2}
\end{bmatrix} + \frac{1}{h} \begin{bmatrix}
0 & 0 & 0 \\
-\frac{a^2}{a+b} & \frac{a^2 + ab + b^2}{a+b} & 0 \\
-\frac{ab}{a+b} & -\frac{b^2}{a+b} & 0
\end{bmatrix}.
\]

The resulting linear system is solved approximately using GMRES(5) (+ preconditioner). We first consider \(h = 1/61\) (i.e. 3600 unknowns).

For several values of \(\varepsilon/h\) the convergence behaviour of GMRES(5) is shown in Figure 1. We clearly see a strong dependence of this convergence behaviour on the parameter \(\varepsilon/h\). In Figure 2 we show the results for GMRES(5) with approximate cyclic reduction preconditioner. In the preconditioned case we clearly have much more robustness with respect to variation in \(\varepsilon/h\). In the preconditioner we use 4 Gauss-Seidel iterations for solving the systems in the red vertices approximately (cf. §3.3). The coarsening method of §3.1 results in \(m_0 = 3600, m_1 = 900, m_2 = 225, m_3 = 64, m_4 = 16\).
In the current approach there appears to be a dependence of the convergence rate on the mesh size $h$. In Figure 3 we show the results for $\epsilon/h = 0.1$ with $h = 1/31$, $h = 1/61$, $h = 1/121$. This is not in agreement with the usual multigrid convergence behaviour. It is not clear yet whether a modified version (e.g. slower coarsening, more accurate inner solvers) exists which does not show this undesirable behaviour.

**Experiment 2.** We take SHERMAN1 from the Harwell-Boeing collection. This is a symmetric matrix of order 1000 with 3750 nonzero entries. The convergence of GMRES(5) is shown in Figure 4. If we use preconditioning we obtain the results shown in Figure 5.
The coarsening method yields $m_0 = 1000$, $m_1 = 346$, $m_2 = 75$, $m_3 = 12$. The total number of nonzero entries in the red vertices systems is 3561. On level 1 we use only 1 Gauss-Seidel iteration in the inner solver; on higher levels we use 4 Gauss-Seidel iterations. This makes the preconditioner very cheap (approximately 3 matrix-vector multiplications).

Experiment 3. We take ORSREG1 from the Harwell-Boeing collection. This is a non-symmetric matrix from oil reservoir simulation of order 2205 with 14133 nonzero entries. The convergence of GMRES(5) is shown in Figure 6. With preconditioning we obtain the results shown in Figure 7. The coarsening method yields $m_0 = 2205$, $m_1 = 1102$, $m_2 = 181$, $m_3 = 29$. On level 1 we use only 1 Gauss-Seidel iteration in the inner solver; on level 2 we use 8 Gauss-Seidel iterations; on level 3 and 4 we use 30 Gauss-Seidel iterations.
References


**PREVIOUS PUBLICATIONS IN THIS SERIES:**

<table>
<thead>
<tr>
<th>Number</th>
<th>Author(s)</th>
<th>Title</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>96-12</td>
<td>S.J.L. van Eijndhoven</td>
<td>Hilding's theorem for Banach spaces</td>
<td>August '96</td>
</tr>
<tr>
<td>96-13</td>
<td>A.F.M. ter Elst, D.W. Robinson</td>
<td>Second-order subelliptic operators on Lie groups I: complex uniformly continuous principal coefficients</td>
<td>August '96</td>
</tr>
<tr>
<td>96-14</td>
<td>A.F.M. ter Elst, D.W. Robinson</td>
<td>High order divergence-form elliptic operators on Lie groups</td>
<td>August '96</td>
</tr>
<tr>
<td>96-15</td>
<td>A.F.M. ter Elst, D.W. Robinson</td>
<td>Second-order subelliptic operators on Lie groups II: real measurable principal coefficients</td>
<td>August '96</td>
</tr>
<tr>
<td>96-16</td>
<td>A.F.M. ter Elst, D.W. Robinson</td>
<td>Second-order strongly elliptic operators on Lie groups with Hölder continuous coefficients</td>
<td>August '96</td>
</tr>
<tr>
<td>96-17</td>
<td>S.L. de Snoo</td>
<td>Boundary conditions for an instationary contact line of a viscous drop spreading on a plate</td>
<td>September '96</td>
</tr>
<tr>
<td>96-18</td>
<td>A.A. Reusken</td>
<td>An approximate cyclic reduction multilevel preconditioner for general sparse matrices</td>
<td>September '96</td>
</tr>
<tr>
<td>96-19</td>
<td>G. Prokert</td>
<td>Existence results for Hele-Shaw flow driven by surface tension</td>
<td>October '96</td>
</tr>
<tr>
<td>96-20</td>
<td>A.A. Reusken</td>
<td>An approximate cyclic reduction multilevel preconditioner for general sparse matrices</td>
<td>October '96</td>
</tr>
</tbody>
</table>