Algebraic Domain Decomposition

Von der Fakultät für Mathematik der Technischen Universität Chemnitz

genehmigte

D i s s e r t a t i o n

zur Erlangung des akademischen Grades

Doctor rerum naturalium

(Dr. rer. nat.)

vorgelegt

von Dipl.–Math. Matthias Bollhöfer

geboren am 14.05.1966 in Herford

eingereicht am 26.11.1997

Gutachter: Prof. Dr. Volker Mehrmann
Prof. Dr. Ludwig Elsner
Prof. Dr. Thomas Huckle

Tag der Verteidigung: 20.03.1998

Bibliographische Beschreibung

Matthias Bollhöfer
Algebraic Domain Decomposition
Dissertation
TU Chemnitz, Fakultät für Mathematik, 1997
211 Seiten, 62 Abbildungen, 135 Tabellen, 82 Literaturverweise

Referat. In der Arbeit wird ein algebraischer Ansatz zur numerischen Lösung großer schwachbesetzter Systeme auf Parallelrechnern diskutiert. Hierbei sollen Techniken aus den Gebietszerlegungsmethoden in der numerischen Behandlung partieller Differentialgleichungen auf den rein algebraischen Fall übertragen werden.
Ausgangspunkt dieser Untersuchungen ist die Sherman–Morrison–Woodbury Formel zur Invertierung von Matrizen bei Modifikationen von niedrigem Rang.

$$(S-W)^{-1} \equiv (S-FG)^{-1} = S^{-1} + S^{-1}F(\underbrace{I-GS^{-1}F}_{S_c})^{-1}GS^{-1}$$

Hauptproblem bei der praktischen Anwendung dieser Formel ist die Lösung eines kleindimensionierten Kopplungssystemes S_c, welches in der Formel auftritt.
Zur numerischen Lösung dieses Kopplungssystems wird ein Konzept eingeführt, welches einen Kompromiß zwischen einer iterativen und einer direkten Lösung von S_c darstellt. Dies geschieht durch geschachtelte Anwendung der Sherman–Morrison–Woodbury Formel. Das führt zu einer Reduktion des Ranges der Restmatrix W in der Anfangszerlegung und somit zur adaptiven Konstruktion eines Vorkonditionieres durch Modifikation der Matrix S. Auf diese Weise sollen die Vorzüge eines iterativen Verfahrens, wie etwa leichte Implementation auf Parallelrechnern, mit der Gewissheit eines direkten Verfahrens, nämlich nach einer bestimmten Anzahl Schritte das System gelöst zu haben, kombiniert werden.
Es wird der Zusammenhang zu algebraischen Mehrgitterverfahren demonstriert.
Zur parallelen Behandlung werden Zerlegungen mit blockdiagonalen Matrizen S betrachtet. Hier werden für symmetrische Matrizen und M–Matrizen strukturerhaltende Modifikationen der blockdiagonalen Matrix untersucht. Darüber hinaus werden weitergehende Modifikationen untersucht, welche die Eigenschaften des Kopplungssystem verbessern sollen. Dieser Ansatz wird auf approximative Lösungen von algebraischen Riccatigleichungen zurückgeführt.
Zur parallelen Realisierung wird das Konzept addierender und überlappender Vektoren, bekannt aus den Gebietszerlegungsmethoden für partielle Differentialgleichungen auf den algebraischen Fall übertragen. Zur parallelen Behandlung der geschachtelten Anwendung der Sherman–Morrison–Woodbury Formel werden Strategien zur Bündelung der einzelnen Aufdatierungen verfolgt zwecks Reduktion der Kommunikation. Dadurch läßt sich das resultierende Kopplungssystem auf die Behandlung des ursprünglichen Kopplungssytems mit zusätzlichen Modifikationen von niedrigem Rang zurückführen.

Schlagwörter. Sherman–Morrison–Woodbury Formel, Gebietszerlegungsmethoden, algebraische Mehrgitterverfahren, blockdiagonale Zerlegungen, große schwachbesetzte Systeme, parallele Algorithmen, iterative Lösungsverfahren, direkte Lösungsverfahren, adaptive Vorkonditionierer.

Ich möchte mich an dieser Stelle recht herzlich bei meinem Betreuer Prof. Dr. Volker Mehrmann für die Betreuung dieser Arbeit bedanken. Seine Unterstützung sowie seine vielen Anregungen und fruchtbaren Diskussionen mit ihm haben mir sehr bei der Erstellung dieser Arbeit geholfen.

Notation

Unless we need explicitly $\mathbb{R}$ or $\mathbb{C}$, we will use the symbol $\mathbb{F}$, which may be replaced by both, $\mathbb{R}$ or $\mathbb{C}$, i.e. $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$.
We use the * to denote the adjoint operation with respect to a given inner product $(\bullet, \bullet)$. If nothing different is mentioned, we assume that the inner product is the standard inner product. In this case, * is either the transposing operation ^{T} for the real case or the conjugate transposition operator ^{H} for the complex case.
For any pair A, B of $n \times n$ symmetric (Hermitian) matrices we define

$$A \leqslant B :\Longleftrightarrow B - A \text{ positive semidefinite.}$$

For any pair A, B of $n \times n$ real matrices we define

$$A \preceq B :\Longleftrightarrow a_{ij} \leqslant b_{ij}, \quad \text{for all } i, j = 1, \ldots, n.$$

$$A \prec B :\Longleftrightarrow a_{ij} < b_{ij}, \quad \text{for all } i, j = 1, \ldots, n.$$

Analogously $\succeq, \succ$ are defined.

Some further notation:

$\mathrm{M}\,(m \times n, \mathbb{F})$	$m \times n$ matrices with entries in $\mathbb{F}$
$\mathrm{GL}\,(n, \mathbb{F})$	nonsingular $n \times n$ matrices
I, I_n	Identity matrix of order n
$\mathrm{diag}(a_{11}, \ldots, a_{nn})$	diagonal matrix with diagonal entries $a_{11}, \ldots, a_{nn}$
$\mathrm{diag} A$	$(a_{11}, \ldots, a_{nn})$, vector containing the diagonal entries of A
$\left(a_{ij}\right)_{i=1,\ldots,m,\, j=1,\ldots,n}$	entries of the $m \times n$ matrix A
$\left(a_{ij}^{(k)}\right)_{i=1,\ldots,m,\, j=1,\ldots,n}$	entries of the $m \times n$ matrix A_k
$\\|A\\|_p$	$\sup_{x \neq 0} \frac{\\|Ax\\|_p}{\\|x\\|_p}$, induced p–norm
$\\|A\\|_F$	$\sqrt{\sum_{r,s=1}^{n} \lvert a_{rs}\rvert^2}$, Frobenius norm of A
$\mathrm{cond}_p(A)$	$\\|A\\|_p \\|A^{-1}\\|_p$, condition number of A, if A is nonsingular
$\Lambda(A)$	$\{\lambda \in \mathbb{C} : \det(A - \lambda I) = 0\}$, set of all eigenvalues of A
$\mathrm{D}(z, r)$	$\{w \in \mathbb{C} : \lvert z - w\rvert < r\}$, open disc with center z and radius r in $\mathbb{C}$
$\mathcal{C}(\Omega)$	space of continuous functions defined on Ω
$\#S$	number of elements in the set S
$\mathrm{Real}\, z$	real part of the complex number z

Contents

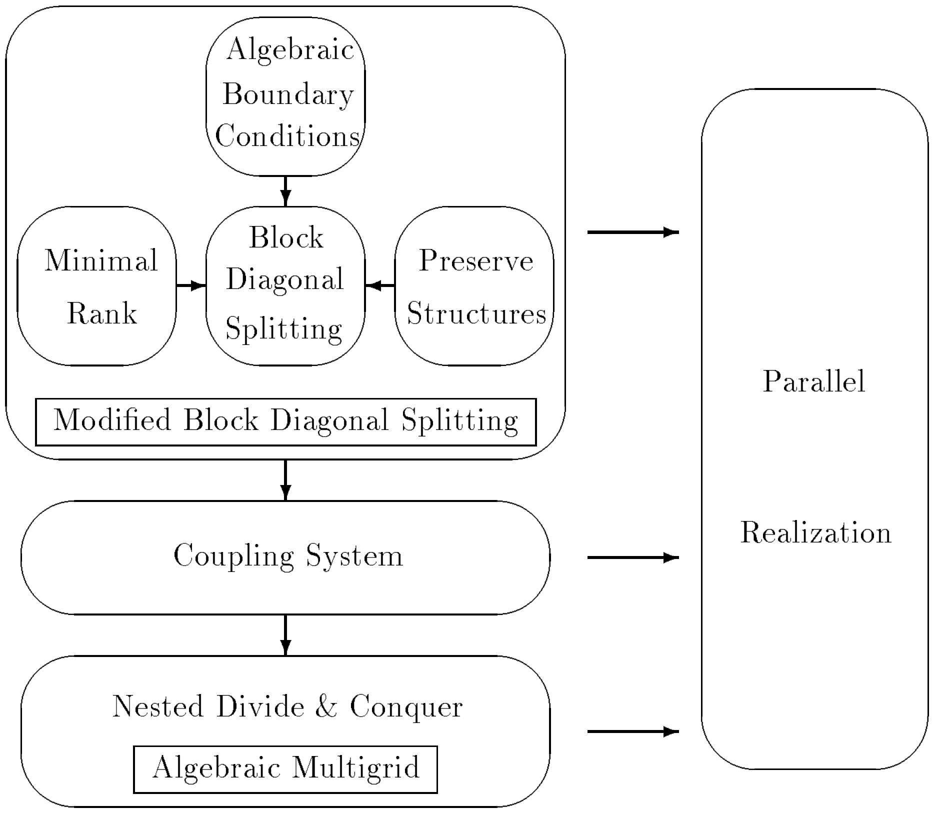
Introduction

In this thesis we will present an algebraic method for the parallel solution of large sparse linear systems. The method is based on the Divide & Conquer approach [12], [59] and uses the Sherman–Morrison–Woodbury formula for low rank modifications. In contrast to domain decomposition methods in the numerical treatment of partial differential equations our method will be an exclusively algebraic domain decomposition method and therefore it can be applied to a large class of problems. The concept of domain decomposition methods for partial differential equations and the techniques for developping parallel algorithms using domain decomposition are transferred to an algebraic method. The method involves the solution of a coupling system of small size, which is involved by applying the Sherman–Morrison–Woodbury formula to a low rank splitting. Most work is spent in how this coupling system can be solved and moreover how to do this in parallel computations. Since it may happen that the coupling system is ill–conditioned, most interest will concentrate on the properties of the coupling system and how to improve the properties. The main idea of improving the properties of the coupling system and the underlying solution process consists of adaptively constructing a preconditioner and reducing the rank of the corresponding coupling system at the same time. Even if an iterative solution process would fail the reduction of the rank will stay and in this case the reduction of the rank will finally lead to a direct method. Of course a purely algebraic method will never be able to compete with a method which has been adapted to a specific related problem. But it may be applied to a wide class of problems by making a compromise between direct and iterative methods. The concept itself invokes several questions and not all of them can be answered in this thesis. Although several parts of this concept can be applied to general systems, here most work is spend in the symmetric positive definite case. For general systems many questions are still open.

For the strategy which will be the topic of this thesis we will give a short overview.
To apply the Sherman–Morrison–Woodbury formula we need a given low rank splitting. Thus we will concentrate on splittings with block diagonal matrices when applying the Sherman–Morrison–Woodbury formula. For this we need a preprocessing process, that suitably partitions the initial system. This step will be assumed to be done a priori. Based on this assumption, we will modify the diagonal blocks with respect to certain aspects. One aspect will be preserving given structures like positive definiteness and M–matrix property for the block diagonal matrix itself and especially for the coupling system. A further aspect will be a local minimization of the rank of the remaining matrix in order to keep the coupling system small. Beside these modifications we need to modify the diagonal blocks to improve the properties of the initial splitting and the underlying coupling system for the case when the block diagonal part of the initial matrix is ill–conditioned or singular. Analogous to partial differential equations we will discuss modifications of the diagonal blocks which one can view as some kind of algebraic boundary conditions subject to maintain the minimal rank property. So far we still need that the block diagonal part of the initial system is nonsingular. In future works we have to generalize this kind of

algebraic boundary conditions.
The coupling system which is obtained by the Sherman–Morrison–Woodbury formula, has to be solved. Circumstances like the distribution of the coupling system over the processors in parallel computations aggravate the solution process. Thus we need a concept for the solution of the coupling system which combines the advantages of an iterative solution process with the certainty of a direct solution process. To do this a nested divide & conquer strategy will be introduced. This strategy is a compromise between a direct solution and an iterative solution of the coupling system. The main idea is to solve some part of the coupling system directly and to obtain a new reduced remaining coupling system. To split the coupling system into a small part and a remaining part, orthogonal transformations are used. While in the symmetric case we can determine orthogonal transformations which optimally reduce the coupling system in the sense of quadratic forms in general it is open which orthogonal transformation will be most suitable. For the nested divide & conquer strategy close relations to algebraic multigrid methods will be shown.
For the parallel realization of this method we need a parallel model for the treatment of the initial coupling system from the Sherman–Morrison–Woodbury formula and moreover for the nested application of this formula in the divide & conquer process. We will present a parallel concept which discusses the parallel use of the Sherman–Morrison–Woodbury formula. By transferring the idea of adding type vectors and overlapping type vectors, which has already been used in domain decomposition methods, we will get a convenient way to treat the coupling system in parallel. This idea can be generalized to the case when the nested divide & conquer process is used. In addition it will be shown how the nested divide & conquer method can be treated in parallel without having too much data traffic.

The complete concept can be summarized in the following table.



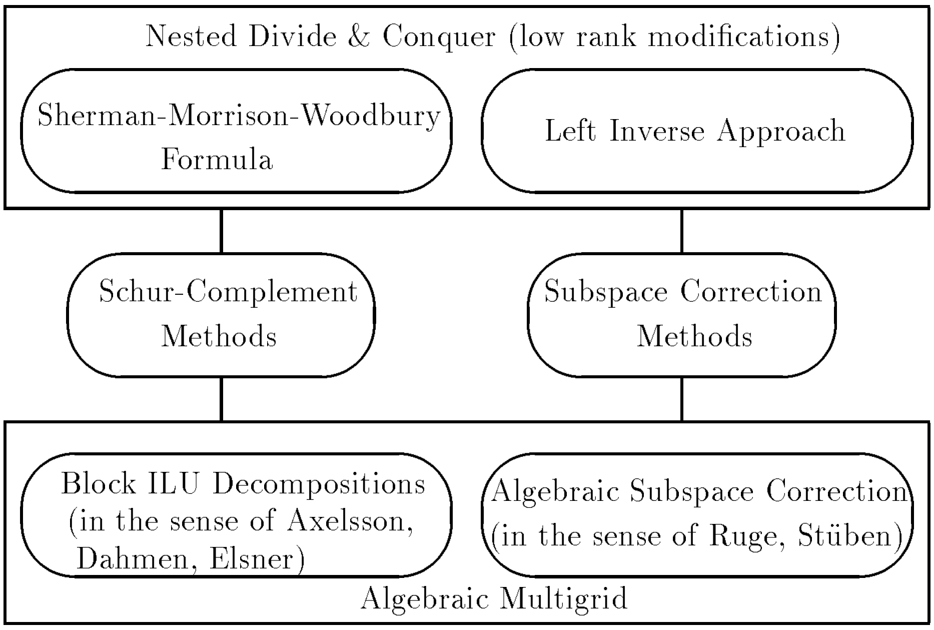
In Chapter 1 we will recall methods of domain decomposition for partial differential equations. Especially substructuring methods are shortly discussed. Analogous to domain decomposition methods we will introduce two variants of algebraic domain decomposition based on low rank modification formulas for a given splitting $A = S - W$, where W has low rank. It will be shown that in theory both methods are equivalent.

In Chapter 2 the properties of the coupling system will be discussed. It will be shown, that the coupling system can be interpreted as the restriction of AS^{-1} to a special invariant subspace. Analogous to Schur–complements for substructuring methods, we will show that structures like symmetry, positive definiteness, the M–Matrix property, the symmetric M–Matrix property are inherited by the coupling system if the initial splitting is suitably modified.

In Chapter 3 we will introduce a nested divide & conquer strategy to improve the properties of the coupling system. This will be an alternative to the usual way which consists in constructing preconditioners. The nested use of low rank modification formula will result in a nested sequence of splittings and consequently a sequence of coupling systems. It will turn out that the related coupling systems can be viewed as diagonal blocks of a block LU–decomposition of the initial coupling system after a suitable pre– and post multiplication.

In Chapter 4 we will point out the close relations to algebraic multigrid methods. It will be shown that algebraic domain decomposition can be interpreted as Schur–complement approach with respect to a suitably extended system. Therefore results from substructuring methods are applicable. A further interpretation as subspace correction method will be pointed out and results from algebraic multigrid methods [72] can be applied.

The relation between the nested use of the low rank modification formulas and algebraic multigrid can be summarized in the following table.

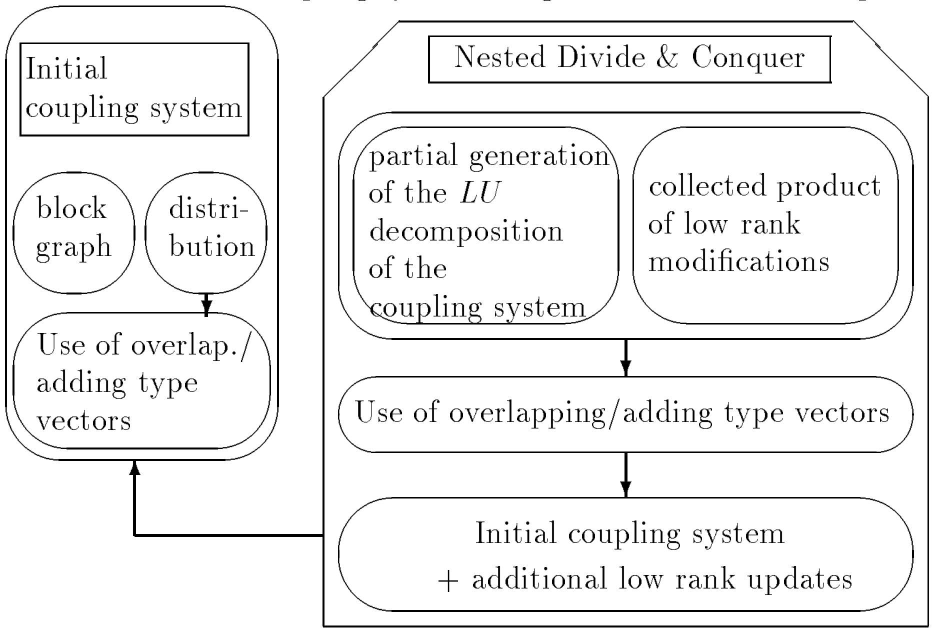

Nested Divide & Conquer (low rank modifications)
Sherman-Morrison-Woodbury Formula
Left Inverse Approach
Schur-Complement Methods
Subspace Correction Methods
Block ILU Decompositions (in the sense of Axelsson, Dahmen, Elsner)
Algebraic Subspace Correction (in the sense of Ruge, Stüben)
Algebraic Multigrid

In Chapter 5 we will discuss block Jacobi splittings and moreover modified block Jacobi splittings. The latter are studied more precisely to construct a factorization of the remaining low rank part. Modifications of the diagonal blocks are applied to inherit structures.

In Chapter 6 we will focus on modifications of block Jacobi–like splittings subject to minimize the rank and to improve the properties of the coupling system. It will be shown that this problem can be traced back to the solution of algebraic Riccati–equations. It will be shown that under relatively general assumptions the Riccati–equation will have explicit solutions. Detailed discussion is carried out for the case that the quadratic part of the algebraic Riccati–expression is nonsingular. For certain classes of matrices it will be shown that this is a realistic assumption. For the special case of symmetric positive definite matrices optimality will be discussed.

In Chapter 7 parallel aspects will be discussed for the coupling system obtained by the Sherman–Morrison–Woodbury formula based on modified block Jacobi splittings.
It will be shown that the coupling system has a natural distribution over the processors. This kind of distribution allows the use of so–called overlapping type vectors and adding type vectors for the parallel treatment of the initial coupling system. Consequently a convenient parallel treatment of the initial coupling system analogous to the numerical treatment of partial differential equations will be possible. The block graph of the initial coupling system and two ways to derive it from the block graph of the initial system will be discussed.

In Chapter 8 parallel aspects are generalized to nested divide & conquer methods based on modified block Jacobi splittings.
For this we use that implicitly a special block LU–decomposition of the coupling system is carried out. Techniques for reducing the data traffic will be discussed which are based on collecting products of low rank modifications to one matrix. Using these techniques the treatment of the coupling system arising from the nested divide & conquer approach can be traced back to the initial coupling system using additional low rank updates.


Initial coupling system
block graph
distri-bution
Use of overlap./ adding type vectors
Nested Divide & Conquer
partial generation of the LU decomposition of the coupling system
collected product of low rank modifications
Use of overlapping/adding type vectors
Initial coupling system + additional low rank updates

In Chapter 9 the theory is illustrated for several numerical examples.

Chapter 1

General Approach

1.1 Domain Decomposition Methods in the Numerical Treatment of Partial Differential Equations

We begin with the description of domain decomposition methods in the numerical treatment of partial differential equations. Consider a linear operator $\mathcal{L}$ from some vector space U into another vector space V. Here U, V are assumed to be suitable subspaces of $\mathcal{C}(\Omega)$, where Ω is a domain in $\mathbb{R}^d$. We denote by $\Gamma = \partial\Omega$ the boundary of the domain Ω. Let $\mathcal{B}$ be a linear operator from space $U\big|_\Gamma$ into a space $W \subseteq \mathcal{C}(\Gamma)$. We consider the following linear problem: Let $f \in V, g \in W$. Find $u \in U$ such that

$$\begin{aligned} \mathcal{L}u &= f \text{ in } \Omega, \\ \mathcal{B}u &= g \text{ on } \Gamma. \end{aligned}$$

In the following we will assume that u is uniquely determined by these equations. In the numerical treatment of these equations the continuous domain Ω is replaced by a finite union of polygonal domains $\Omega^h = \bigcup_{i\in I} T_i^h$, where h is some discretization parameter corresponding to the largest diameter of $T_i^h, i \in I$. Γ is replaced by $\Gamma^h = \partial\Omega^h$, where $\Gamma^h = \bigcup_{j\in J} s_j^h$ is defined with respect to Ω^h. $\{s_j^h : j \in J\}$ is the set of edges corresponding to those T_i^h which intersect with $\partial\Omega^h$. The spaces U, V are replaced by some appropriate finite dimensional subspaces U^h, V^h. Typically $U^h\big|_{T_i^h}, V^h\big|_{T_i^h}$ are subspaces of the space of polynomials of degree k on T_i^h for some small fixed k, e.g. $k = 1$. Analogously $W^h\big|_{s_i^h}$ is a subspace of the space of polynomials of degree k on s_i^h. Since U^h, V^h, W^h are finite dimensional, we can find some basis for each space. In addition each basis function u_i in U should have a local support, i.e., $\operatorname{supp} u_i \subseteq T_{i_1} \cup \ldots \cup T_{i_l}$, where l is fixed, $l \ll \#I$ and $u_i(x) = 0$ if $x \notin \operatorname{supp} u_i$. We can thus replace the continuous problem by a discrete problem: Let $f \in V^h, g \in W^h$. Find $u \in U^h$ such that

$$\begin{aligned} \mathcal{L}^h u &= f \text{ in } \Omega^h, \\ \mathcal{B}^h u &= g \text{ on } \Gamma^h, \end{aligned}$$

where $\mathcal{L}^h, \mathcal{B}^h$ are some discrete approximations to $\mathcal{L}, \mathcal{B}$. Again we will assume that u is uniquely determined by these equations. For simplicity let $\dim U^h = n, \dim V^h = m$ and $\dim W^h = n - m$. Since we have a representation $u = \sum_{i=1}^n x_i u_i,\ f = \sum_{i=1}^m b_i v_i,\ g = \sum_{i=1}^{n-m} c_i w_i$ for base $u_1, \dots, u_n$ of U^h, $v_1, \dots, v_m$ of V^h and $w_1, \dots, w_{n-m}$ of W^h, where each u_i has a local support, we can reformulate this problem in terms of coordinates as

$$Ax \equiv \begin{pmatrix} L \\ B \end{pmatrix} x = \begin{pmatrix} b \\ c \end{pmatrix}$$

where $\mathcal{L}^h u_j = \sum_{i=1}^m l_{ij} v_i$, $\mathcal{B}^h u_j = \sum_{k=1}^{n-m} b_{kj} w_k$.
For the coefficients l_{ij}, b_{kj} we will assume that $l_{ij} = 0$ implies that $\operatorname{supp} u_i \cap \operatorname{supp} u_j$ has zero d–dimensional measure and $b_{kj} = 0$ implies that $\operatorname{supp} u_{m+k} \cap \operatorname{supp} u_j$ has zero d–dimensional measure. This is a typical property of problems arising from partial differential equations. By theses assumptions we obtain a nonsingular sparse matrix A. Moreover, the sparsity pattern is essentially determined from the discretization of the domain and the choice of the local support for the basis functions. Since A is nonsingular, it follows that B must have full rank. Thus we can find $n - m$ linear independent columns in B. Without loss of generality we can assume that the last $n-m$ columns are linear independent. We partition $L = (L_I, L_{I,\Gamma}), B = (B_{\Gamma,I}, B_\Gamma)$ and $x = \binom{x_I}{x_\Gamma}$ such that B_Γ is nonsingular. Then we can reduce the system

$$\begin{pmatrix} L_I & L_{I,\Gamma} \\ B_{\Gamma,I} & B_\Gamma \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}$$

to

$$\underbrace{\left(L_I - L_{I,\Gamma} B_\Gamma^{-1} B_{\Gamma,I}\right)}_{K} x_I = \underbrace{b - L_{I,\Gamma} B_\Gamma^{-1} c}_{d},$$

which is the Schur–complement of A with respect to B_Γ. We set $K = L_I - L_{I,\Gamma} B_\Gamma^{-1} B_{\Gamma,I},\ d = b - L_{I,\Gamma} B_\Gamma^{-1} c$. Usually one is not interested in x_Γ, so that it suffices to solve the last system. Note that the Schur–complement K differs from L_I at most in those entries k_{ij}, where $\operatorname{supp} u_i$ and $\operatorname{supp} u_j$ have a nontrivial intersection with $\bigcup_{k=m+1,\dots,n} \operatorname{supp} u_k$, which is the common support of those basis functions which correspond to B_Γ. This can be seen by an argument from graph theory [27].

In general there are two approaches of domain decomposition methods to problems of this type.

1.1.1 Substructuring Methods

The first class of techniques are called substructuring methods. For such methods Ω^h is subdivided into subdomains $\Omega_1^h, \dots, \Omega_p^h$ which are simply connected with respect to Ω^h and $\Omega_i^h \cap \Omega_j^h$ has zero d–dimensional measure if $i \neq j$. On each subdomain one has to consider the corresponding smaller dimensional problem. But usually the problem is no longer uniquely determined due to the additional boundary between neighbouring subdomains. For any $r = 1, \dots, p$ we denote by $u_1^{(r)}, \dots, u_{k_r}^{(r)}$ those basis functions which have a nontrivial support in $\Omega_r^h,\ r = 1, \dots, p$. On each subdomain we consider the corresponding problem: Find

$u \in \operatorname{span}\{u_1^{(r)}, \ldots, u_{k_r}^{(r)}\}$ such that

$$\begin{aligned}
\mathcal{L}^h u &= f \text{ in } \Omega_r^h,\\
\mathcal{B}^h u &= g \text{ on } \Gamma^h \cap \partial\Omega_r^h,\\
\mathcal{C}^h u &= 0 \text{ on } \partial\Omega_r^h \setminus \Gamma^h
\end{aligned}$$

The additional boundary condition $\mathcal{C}^h u = 0$ has been added to the problem in order to obtain a unique solution. The number of equations, which have to be added is just the number of basis functions in $\{u_1^{(r)}, \ldots, u_{k_r}^{(r)}\}$ whose support is not covered by Ω_r^h. For those basis functions, which have a support in more than a single subdomain, we have to consider an additional coupling system. For the discrete system $Kx_I = d$ this means, that we have up to a permutation P

$$K = P \begin{pmatrix} K_{1,1} & & & K_{1,p+1} \\ & \ddots & & \vdots \\ & & K_{p,p} & K_{p,p+1} \\ K_{p+1,1} & \cdots & K_{p+1,p} & K_{p+1,p+1} \end{pmatrix} P^T,$$

where each $K_{ii},\ i = 1, \ldots, p$ corresponds to the discrete problem for the basis functions with support covered by Ω_i^h. Together with the additional boundary conditions of the form $\left(C_{p+1,i}^{(i)},\ C_{p+1,p+1}^{(i)}\right)\binom{u}{v} = 0$ we obtain subproblems of the form

$$\begin{pmatrix} K_{ii} & K_{i,p+1} \\ C_{p+1,i}^{(i)} & C_{p+1,p+1}^{(i)} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} d_i \\ 0 \end{pmatrix}.$$

Applying the Schur–complement gives

$$\hat{K}_{ii} u \equiv \left(K_{ii} - K_{i,p+1} {C_{p+1,p+1}^{(i)}}^{-1} C_{p+1,i}^{(i)}\right) u = d_i$$

From this it follows that we have to solve a system with $\hat{K}$ instead of K, where $\hat{K}$ is defined

$$\hat{K} := K - \begin{pmatrix} K_{1,p+1} & & \\ & \ddots & \\ & & K_{p,p+1} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} \end{pmatrix} \begin{pmatrix} C_{p+1,p+1}^{(1)} & & \\ & \ddots & \\ & & C_{p+1,p+1}^{(p)} \end{pmatrix}^{-1} \begin{pmatrix} C_{p+1,1}^{(1)} & & & \mathrm{O} \\ & \ddots & & \mathrm{O} \\ & & C_{p+1,p}^{(p)} & \mathrm{O} \end{pmatrix}$$

For problems which ensure that each principal submatrix is nonsingular like positive definite matrices, M–matrices or diagonal dominant matrices we can choose $\left(C_{p+1,i}^{(i)},\ C_{p+1,p+1}^{(i)}\right) = (\mathrm{O}, I)$. In this case we obviously have $\hat{K} = K$. Otherwise $\hat{K}$ differs from K by a low rank modification. The rank is related to the basis functions with support in more than a single subdomain. A system with $\hat{K}$ can be solved using the Schur–complement approach, i.e., using the following factorization for $\hat{K}$

$$\begin{pmatrix} I & & & \\ & \ddots & & \\ & & I & \\ K_{p+1,1}\hat{K}_{11}^{-1} & \cdots & K_{p+1,p}\hat{K}_{pp}^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{K}_{11} & & & \\ & \ddots & & \\ & & \hat{K}_{pp} & \\ & & & C_{p+1,p+1} \end{pmatrix} \begin{pmatrix} I & & & \hat{K}_{11}^{-1}K_{1,p+1} \\ & \ddots & & \vdots \\ & & I & \hat{K}_{pp}^{-1}K_{p,p+1} \\ & & & I \end{pmatrix},$$

where $C_{p+1,p+1} = C_{p+1,p+1} - \sum_{i=1}^{p} K_{p+1,i}\hat{K}_{ii}^{-1}K_{i,p+1}$ denotes the Schur–complement. Unless $\hat{K} = K$ one has to solve an additional coupling system in order to obtain the solution of $Kx_I = d$. For the solution of systems by the Schur–Complement approach most interest is focussed on solving the Schur–Complement system $C_{p+1,p+1}$, since usually $C_{p+1,p+1}$ is not computed explicitly. For this purpose Krylov subspace based methods [41] are used, which only need matrix vector multiplications. In order to accelerate the iteration process preconditioners are constructed for $C_{p+1,p+1}$. For the case of elliptic boundary value problems efficient preconditioners have been developed in [9], [24],[15]. Another way to solve the system $Kx_I = d$ is to use an approximate factorization for $\hat{K}$, i.e. a factorization where each $\hat{K}_{ii}^{-1}$ is replaced by a suitable approximation. This incomplete factorization can be used as preconditioner for K. Again it is necessary to solve an approximate Schur–complement in each step, where one can use the preconditioners just mentioned. Such methods have been proposed by [45], [46].

1.1.2 Overlapping Domain Decomposition Methods

A second class of domain decomposition methods is given by overlapping or additive Schwarz methods. For these methods the domain Ω is again replaced by Ω^H, which is a union of "coarse" polygons T_i^H. But this time each polygon T_i^H is refined into a union of smaller polygons T_{ij}^h, $j = 1, \ldots, l_i$. The initial decomposition of Ω^H is given by the coarse polygons T_i^H. Then each subdomain T_i^H is enlarged by some neighbouring polygons T_{kj}^h resulting in a subdomain $\hat{T}_i^H$. A basis $u_1^h, \ldots u_n^h$ with local support of the refined domain can be constructed with respect to the polygons T_{ij}^h and an additional basis $u_1^H, \ldots, u_k^H$ with local support can be constructed for the coarse space Ω^H. By construction we have $\operatorname{span}\{u_1^H, \ldots, u_k^H\} \subseteq \operatorname{span}\{u_1^h, \ldots, u_n^h\}$. The main difference to substructuring methods is that here the subdomains $\hat{T}_i^H$ have to be sufficiently large in order to ensure that for any u_r there exists at least one subdomain $\hat{T}_i^H$ such that $\operatorname{supp} u_r \subseteq \hat{T}_i^H$. For any r we denote by $u_1^{(r)}, \ldots, u_{k_r}^{(r)}$ the basis functions, which have support in $\hat{T}_r^H$. For each subdomain $\hat{T}_r^H$ one has to consider the corresponding subproblem. Find $u \in \operatorname{span}\{u_1^{(r)}, \ldots, u_{k_r}^{(r)}\}$ such that

$$\begin{aligned} \mathcal{L}^h u &= f \text{ in } \hat{T}_r^H, \\ \mathcal{B}^h u &= g \text{ on } \Gamma^h \cap \partial\hat{T}_r^H, \\ \mathcal{C}^h u &= 0 \text{ on } \partial\hat{T}_r^H \setminus \Gamma^h \end{aligned}$$

In addition one may consider the coarse problem. Find $u \in \operatorname{span}\{u_1^H, \ldots, u_k^H\}$ such that

$$\begin{aligned} \mathcal{L}^H u &= f \text{ in } \Omega^H, \\ \mathcal{B}^H u &= g \text{ on } \Gamma^H. \end{aligned}$$

We can write the reduced matrix K as

$$K = P_r \begin{pmatrix} K_{11}^{(r)} & K_{12}^{(r)} \\ K_{21}^{(r)} & K_{22}^{(r)} \end{pmatrix} P_r^T,$$

where P_r is a permutation matrix and $K_{11}^{(r)}$ corresponds to the subproblem for those basis functions which have support in $\hat{T}_r^H$. Analogous to substructuring methods we can rewrite the subproblems on $\hat{T}_r^H$ in the form

$$\hat{K}_{11}^{(r)} u \equiv (K_{11}^{(r)} - K_{12}^{(r)} C_{22}^{(r)^{-1}} C_{21}^{(r)}) u = d_r,$$

where $\left(C_{21}^{(r)}, C_{22}^{(r)}\right)$ corresponds to the additional boundary condition $\mathcal{C}^h u = 0$. Again $\hat{K}_{11}^{(r)}$ differs from $K_{11}^{(r)}$ only by a low rank modification, which is related to the basis functions with support in more than one subdomain. By the choice of T_i^h we can already build a complete approximate solution to $Kx_I = d$ from these subproblems.
For the coarse problem we can find a matrix G such that $\left(u_1^H, \ldots, u_k^H\right) = \left(u_1^h, \ldots, u_n^h\right) G$. The corresponding linear operator is given by

$$\hat{K} = G(G^T K G)^{-1} G^T + \sum_{r=1}^{p} P_r \begin{pmatrix} (\hat{K}_{11}^{(r)})^{-1} & 0 \\ 0 & 0 \end{pmatrix} P_r^T.$$

This matrix is used as preconditioner for K in Krylov subspace based methods. It has been shown in [25], [26] that for elliptic boundary value problems with shape regular finite element discretization, $\frac{\lambda_{\max}(\hat{K}^{-1}K)}{\lambda_{\min}(\hat{K}^{-1}K)}$ has a condition number independent on h, H and p, provided that the overlap size of $\hat{T}_r^H$ with respect to T_r^H is bounded from below by fixed fraction of H. If the expression $G(G^T K G)^{-1} G^T$ is omitted, then $\frac{\lambda_{\max}(\hat{K}^{-1}K)}{\lambda_{\min}(\hat{K}^{-1}K)}$ grows at least as fast as $1/H^2$, which has been shown in [83].

1.2 Algebraic Domain Decomposition

Consider a large sparse nonsingular matrix $A \in \mathrm{GL}\,(n, \mathbb{F})$ as it occurs in the numerical treatment of partial differential equations (see e.g. [48]). Typically the sparsity can be characterized as follows: In each row there are only a small number of nonzero entries and the pattern of the matrix is almost symmetric. The matrix can be permuted by a symmetric permutation into a form which is almost block tridiagonal or block cyclic with blocks of moderate size. Although we do not need this property explicitly, the method will be designed for such matrices.
Now we will discuss algebraic domain decomposition, which is based on a low rank modification formula. For an algebraic method the linear system is the only information we have. For domain decomposition methods arising from partial differential the decomposition is constructed by geometric aspects. Here the domain decomposition will be done partitioning the vector of unknowns according to some strategy which should be related to the undirected graph of A, where the graph $G(A) = (\mathcal{V}, \mathcal{E})$ of A is defined as follows:

$$\mathcal{V} = \{1, \ldots, n\}, \ \mathcal{E} = \{(i,j) \in \mathcal{V} \times \mathcal{V} : a_{ij} \neq 0 \text{ or } a_{ji} \neq 0\}$$

For strategies of permuting A by renumbering the nodes of $\mathcal{V}$ we refer to [52], [17], [34], [70], [80]. This preprocessing part is assumed to be done a priori.

We wish to solve the linear system

$$Ax = b, \tag{1.1}$$

where $x, b \in \mathbb{F}^n$, on a parallel computer. This is done by splitting the matrix A into a sum of two matrices $S \in \mathrm{GL}\,(n, \mathbb{F}), W \in \mathrm{M}\,(n \times n, \mathbb{F})$, such that

$$A = S - W, \tag{1.2}$$

where S is nonsingular, the solution of a linear system for S can be "easily" done in parallel and W is a matrix of low rank. The most common choice for S will be a block diagonal matrix with as many diagonal blocks as processors. But so far we will not fix this explicitly. For W we assume more precisely, that W is a product of matrices $W = FX^{-1}G$, where $F, G^T \in \mathrm{M}\,(n \times r, \mathbb{F}), X \in \mathrm{GL}\,(r, \mathbb{F})$ with suitable $r \in \mathbb{N}$, $r \ll n$.

Example 1.3 *Consider a block tridiagonal matrix $A \in \mathrm{GL}(n, \mathbb{R})$, where*

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & & & \\ A_{2,1} & A_{2,2} & A_{2,3} & & \\ & \ddots & \ddots & \ddots & \\ & & A_{m-1,m-2} & A_{m-1,m-1} & A_{m-1,m} \\ & & & A_{m,m-1} & A_{m,m} \end{pmatrix}$$

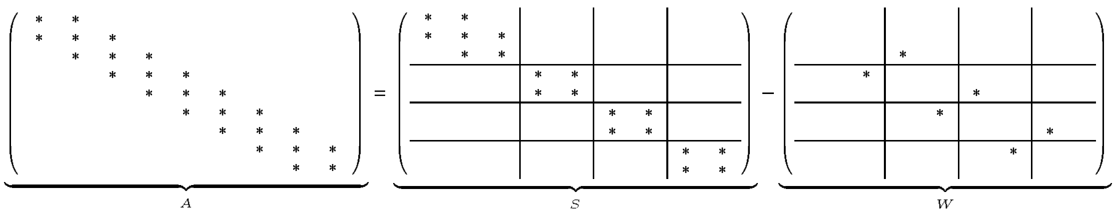
and the $A_{i,j}$'s are matrices of suitable size. Assume that each principal submatrix is invertible.(This condition is fulfilled for example if A is strictly diagonal dominant, symmetric positive definite or an M–Matrix). Split A into a block diagonal matrix S with p diagonal blocks and remaining W, where for suitable $m_1, \dots, m_{p-1} \in \{1, \dots, m-1\}, m_0 := 0, m_p := m$

$$S = \begin{pmatrix} S_{1,1} & & \\ & \ddots & \\ & & S_{p,p} \end{pmatrix}$$

and for $j = 1, \dots, p$ the diagonal blocks of S are

$$S_j = \begin{pmatrix} A_{m_{j-1}+1,m_{j-1}+1} & A_{m_{j-1}+1,m_{j-1}+2} & & \\ A_{m_{j-1}+2,m_{j-1}+1} & A_{m_{j-1}+2,m_{j-1}+2} & A_{m_{j-1}+2,m_{j-1}+3} & \\ & \ddots & \ddots & \ddots \\ & & A_{m_j,m_j-1} & A_{m_j,m_j} \end{pmatrix}$$

and $W = S - A$ is of low rank. This splitting is illustrated in the figure below:



If A arises from the numerical treatment of partial differential equations, then the block diagonal part S can be interpreted as domain decomposition of the problem. W corresponds to the connections between neighbouring domains.
Set $F := \left(E_{m_1}, E_{m_1+1}, E_{m_2}, E_{m_2+1}, \ldots, E_{m_{p-1}}, E_{m_{p-1}+1}\right)$, where the identity matrix I is partitioned in block columns analogous to the block structure of A, i.e., $I = (E_1, E_2, \ldots, E_m)$. Then W may be written in the following way:

$$\tilde{W} := \begin{pmatrix} \boxed{\begin{array}{c|c} 0 & -A_{m_1,m_1+1} \\ \hline -A_{m_1+1,m_1} & 0 \end{array}} & & \\ & \ddots & \\ & & \boxed{\begin{array}{c|c} 0 & -A_{m_{p-1},m_{p-1}+1} \\ \hline -A_{m_{p-1}+1,m_{p-1}} & 0 \end{array}} \end{pmatrix},$$

$$G := \tilde{W}F^*,$$
$$\Longrightarrow W = FG = F(\tilde{W}F^*).$$

Obviously F has full rank and its number of columns is for $(p \ll m)$ much less than n. Here we have chosen $X = I$.

We may solve a linear system of this form (1.2) using the Sherman–Morrison–Woodbury formula [41], p.51:

(1.4) $\quad A^{-1} = (S - FX^{-1}G)^{-1} = S^{-1} + S^{-1}FS_c^{-1}GS^{-1}$, where $S_c = X - GS^{-1}F$.

S_c is called the <u>**coupling system**</u>. We note that if S and X are nonsingular, then S_c is nonsingular if and only if A is nonsingular.
We get the following abstract algorithm:

Algorithm 1.5
Let $A \in \mathrm{GL}(n, \mathbb{F})$ be split as $A = S - FX^{-1}G$,
where $S \in \mathrm{GL}(n, \mathbb{F}), F, G^T \in \mathrm{M}(n \times r, \mathbb{F}), X \in \mathrm{GL}(r, \mathbb{F})$.

```
Solve $S\psi = b$
$\rho := G\psi$
Solve $S_c\delta = \rho$, where $S_c = X - GS^{-1}F$
$u := F\delta$
Solve $S\gamma = u$
$x := \gamma + \psi$
$\Longrightarrow x \equiv A^{-1}b$
```

In general, we may choose $X = I$ in our factorization, but there may still be difficulties in the practical implementation of this algorithm.
First of all, we have to solve a system with S. If we use a direct method, like LU or

QR decompositions [41],pp.92ff, pp.211ff, we need most of the computing time for the decomposition, while solving several systems with S is not so expensive. For the coupling system we have to decide, whether we want to use a direct or iterative method. At least if we use an iterative method we should ensure, that we only have to apply matrix vector operations, because the explicit computation of S_c may be expensive. For a direct method in general, we have to compute $S^{-1}F$ explicitly and then to use a method which requires explicit knowledge of S_c. Such an approach was performed for block tridiagonal matrices in [12], [59]. In addition, in [59] the rank of F has been chosen as small as possible, to decrease the computational effort. For matrices arising from the numerical treatment of partial differential equations such a strategy corresponds to a special choice of boundary conditions on each subdomain.

1.3 An Equivalent Approach for the Use of Low Rank Modifications

In this section we will derive an equivalent way to write the inverse of a matrix A subject to low rank modifications. This formula will later be used to get another interpretation of using low rank modifications.
For this we have to assume that $X = I$ in our low rank splitting (1.4), i.e.

$$A = S - FG, \tag{1.6}$$

where $S \in \mathrm{GL}\,(n, \mathbb{F})$, $F, G^T \in \mathrm{M}\,(n \times r, \mathbb{F})$ and at least F must have full rank.

In this case there exists $H \in \mathrm{M}\,(r \times n, \mathbb{F})$ such that

$$HF = I_r \tag{1.7}$$

and we obtain

$$GS^{-1} = H(I - AS^{-1}), \qquad S_c = I - GS^{-1}F = HAS^{-1}F =: T_c. \tag{1.8}$$

T_c is also called a <u>**coupling system**</u>. Thus we can write the inverse of A slightly different than in the previous section as

$$A^{-1} = S^{-1} + S^{-1}FT_c^{-1}H(I - AS^{-1}). \tag{1.9}$$

It is easy to verify by straightforward computation that this formula is equivalent to the Sherman–Morrison–Woodbury formula (1.4) from the previous section, provided that F has full rank. Moreover, in this case the coupling systems T_c, S_c are identical.

Example 1.10 *Consider the matrix from Example 1.3. In this case we can choose $H = F^*$.*

1.4 Approximate Inverses

In this section we consider the case, when in formula (1.4),(1.9) the exact inverse S^{-1} is replaced by an appropriate approximation $\tilde{S}^{-1}$. We state the result as a lemma:

Lemma 1.11
Let $A \in \mathrm{GL}(n, \mathbb{F})$ be split as $A = S - FG$, where $S \in \mathrm{GL}(n, \mathbb{F})$, $F, G^T \in \mathrm{M}(n \times r, \mathbb{F})$. Consider an $\tilde{S} \in \mathrm{GL}(n, \mathbb{F})$ such that $A - (S - \tilde{S})$ is still nonsingular. Then

$$\tilde{S}^{-1} + \tilde{S}^{-1} F \tilde{S}_c^{-1} G \tilde{S}^{-1} = [A - (S - \tilde{S})]^{-1}, \tag{1.12}$$

where $\tilde{S}_c = I - G\tilde{S}^{-1}F$.
Assume that $\operatorname{rank} F = r$ and let $H \in \mathrm{M}(r \times n, \mathbb{F})$ such that $HF = I$. Then

$$\tilde{S}^{-1} + \tilde{S}^{-1} F \tilde{T}_c^{-1} H(I - A\tilde{S}^{-1}) = [A - (I - FH)(S - \tilde{S})]^{-1}, \tag{1.13}$$

where $\tilde{T}_c = HA\tilde{S}^{-1}F$.

Proof:
(1.12) follows directly from applying the Sherman–Morrison–Woodbury formula (1.4) to $\tilde{A} = \tilde{S} - FG$.

If we apply (1.9) to $\hat{A} = A - (I - FH)(S - \tilde{S}) = \tilde{S} - F(G + H(\tilde{S} - S))$, we get

$$\begin{aligned} \hat{A}^{-1} &= \tilde{S}^{-1} + \tilde{S}^{-1} F (H\hat{A}\tilde{S}^{-1}F)^{-1} H(I - \hat{A}\tilde{S}^{-1}) \\ &= \tilde{S}^{-1} + \tilde{S}^{-1} F (H\hat{A}\tilde{S}^{-1}F)^{-1} H(I - \hat{A}\tilde{S}^{-1}). \end{aligned}$$

We have that

$$\begin{aligned} H\hat{A} &= H(\tilde{S} - F(G + H(\tilde{S} - S))) \\ &= H\tilde{S} - G - H(\tilde{S} - S) \\ &= HS - G \\ &= H(S - FG) \\ &= HA. \end{aligned}$$

From this it follows that

$$H\hat{A}\tilde{S}^{-1}F = HA\tilde{S}^{-1}F$$

and

$$\begin{aligned} H(I - \hat{A}\tilde{S}^{-1}) &= H - H\hat{A}\tilde{S}^{-1} \\ &= H - HA\tilde{S}^{-1} \\ &= H(I - A\tilde{S}^{-1}). \end{aligned}$$

□

Remark:
We note that (1.12) can be used to define linear iteration schemes for solving systems with A. Especially (1.13) is of interest, since

$$\begin{aligned} & (I - [A - (I - FH)(S - \tilde{S})]^{-1}A) \\ (1.14) \qquad = \; & (I - \tilde{S}^{-1}F\tilde{T}_c^{-1}HA)(I - \tilde{S}^{-1}A), \text{ where } \tilde{T}_c = HA\tilde{S}^{-1}F. \end{aligned}$$

This means, if we use (1.13) for linear iteration schemes, then the corresponding iteration operator decouples into a product of two iteration operators.
Note that one can use analogous arguments if G has full rank instead of F.

Summary

Analogous to domain decomposition methods we have introduced an algebraic way of domain decomposition, where the decomposition is done with respect to low rank modifications. Both approaches involve the solution of a small coupling system.

The Sherman–Morrison–Woodbury formula (1.4) will play a central role in our forthcoming investigations while the equivalent approach (1.9) will give another interpretation of using low rank modifications. The formula as it stands now may cause problems in its practical application.

For the concept of algebraic domain decomposition problems may be the following ones.

1. What general properties of the coupling system S_c can be shown and can we suitably modify the initial splitting $A = S - W$ to improve the properties of S_c?

2. Can we modify the initial splitting such that the coupling system inherits structures of the initial system like symmetry, positive definiteness, M–matrix property?

3. Iterative methods applied to the coupling system and especially Krylov subspace methods may fail when being applied to S_c while direct methods require explicit knowledge of S_c. And even if S_c is explicitly available, a direct solution typically will be aggravated by the distribution over the processors. Can we find a compromise between both approaches? I.e., if we need more and more iterations then more and more of the coupling system should already be directly solved leading finally to a direct solution of S_c.

4. Since we are interested in parallel computations we will restrict ourselves to modified block diagonal splittings. When adaptively generating parts of S_c this must be carefully handled in parallel. The question will be how this can be done.

We will try to give an answer to these questions in the following chapters.

The algebraic properties of the coupling system will be discussed in the next chapter.

Chapter 2

Algebraic Properties of the Coupling System

In this chapter we will discuss properties of the system matrix S_c of the coupling system from (1.4).
The coupling system S_c plays a central role in the use of the Sherman–Morrison–Woodbury formula (1.4). We will illustrate how the application of formula (1.4) is connected to the invariant subspace of AS^{-1} spanned by the columns of F. To do this we will apply the formula to a given right hand side b:
$x_0 := S^{-1}b$. Thus we can compute the residual $r = b - Ax_0 = b - AS^{-1}b = F(GS^{-1}b)$. Here we have used the relation $A = S - FG$ from (1.6). After we he have computed x_0 the residual r lies in the subspace spanned by the columns of F. Since F has low rank we are able to compute the desired solution x of $Ax = b$ by solving a system of small size. But the system of small size is S_c since we have $(AS^{-1})F = F\,S_c$. If F has full rank, then S_c is precisely the restriction of AS^{-1} to the invariant subspace of AS^{-1} formed by the columns of F. This indicates the close relation between S_c and AS^{-1}.
We will discuss general properties of S_c and more special properties for some classes of matrices, namely symmetric matrices, symmetric positive definite matrices, M–matrices. Note that for substructuring methods in the numerical treatment of partial differential equations most work is spent on the solution of the Schur–complement. A well–known advantage of the Schur–complement is that it inherits several structures from the initial system like symmetry, positive definiteness, diagonal dominance or the M–matrix property. However, for the coupling system S_c of an algebraic domain decomposition we will show in this chapter, that we have to construct the splitting and the factorization $W = FX^{-1}G$ carefully to obtain analogous results. For this we have to examine the properties of the coupling system and its relation to the splitting $A = S - W$.

2.1 General Properties

Lemma 2.1 *Let $A, B \in \mathrm{M}(n \times r, \mathbb{C})$. Assume that $n \geqslant r$.*

(i) There exist nonsingular matrices $Y_1, Y_2 \in \mathrm{GL}\,(n,\mathbb{C})$ *such that*

$$Y_1^{-1}AB^*Y_1 = \begin{pmatrix} J & \mathrm{O} \\ \mathrm{O} & N_1 \end{pmatrix}, \quad Y_2B^*AY_2^{-1} = \begin{pmatrix} J & \mathrm{O} \\ \mathrm{O} & N_2 \end{pmatrix},$$

are both in Jordan canonical form and N_1 *and* N_2 *are nilpotent. If* s, t *are the smallest numbers such that* $N_1^s = \mathrm{O}, N_2^t = \mathrm{O}$*, then* $|s-t| \leqslant 1$*.*

(ii) If A *has full rank, then there exists a nonsingular matrix* $Y \in \mathrm{GL}\,(n,\mathbb{C})$ *such that*

$$Y^{-1}(AB^*)Y = \begin{pmatrix} B^*A & * \\ \mathrm{O} & \mathrm{O} \end{pmatrix},$$

where the first r *columns of* Y *are those of* A*.*

Proof:
For the first assertion we make a full rank decomposition of A:

$$A = FG^*,$$

where $F \in \mathrm{M}(n \times l, \mathbb{F})$, $G \in \mathrm{M}(r \times l, \mathbb{F})$, $l = \operatorname{rank} A$. If A has already full rank, then we choose $F = A$, $G = I$. We can always find $H \in \mathrm{M}\,(n \times (n-l), \mathbb{F})$ in such a way that

$$\tilde{Y}_1 = (F, H)$$

is nonsingular. Then we partition

$$\tilde{Y}_1^{-1} = \begin{pmatrix} \hat{F}^* \\ \hat{H}^* \end{pmatrix}.$$

From this it follows, that

$$\begin{aligned} \tilde{Y}_1^{-1}AB^*\tilde{Y}_1 &= \begin{pmatrix} \hat{F}^*F \\ \hat{H}^*F \end{pmatrix} G^*B^*Y_1 \\ &= \begin{pmatrix} I \\ \mathrm{O} \end{pmatrix} G^*B^*Y_1 \\ &= \begin{pmatrix} I \\ \mathrm{O} \end{pmatrix} (G^*B^*F, G^*B^*H) \\ &= \begin{pmatrix} G^*B^*F & G^*B^*H \\ \mathrm{O} & \mathrm{O} \end{pmatrix}. \end{aligned} \tag{2.2}$$

Analogously we find a nonsingular $\tilde{Y}_2 = \binom{G^*}{K^*}$ such that

$$\tilde{Y}_2B^*A\tilde{Y}_2^{-1} = \begin{pmatrix} G^*B^*F & \mathrm{O} \\ K^*B^*F & \mathrm{O} \end{pmatrix}. \tag{2.3}$$

We can transform G^*B^*F to Jordan canonical form:

$$G^*B^*F = Z\begin{pmatrix} J & \mathrm{O} \\ \mathrm{O} & N \end{pmatrix}Z^{-1}, \tag{2.4}$$

where N is the nilpotent part. We set

$$Y_1 = Y_1 \begin{pmatrix} Z & -Z \begin{pmatrix} J^{-1} & 0 \\ 0 & 0 \end{pmatrix} Z^{-1} G^* B^* H \\ 0 & I \end{pmatrix}, \; Y_2 = \begin{pmatrix} Z^{-1} & 0 \\ -K^* B^* F Z \begin{pmatrix} J^{-1} & 0 \\ 0 & 0 \end{pmatrix} Z^{-1} & I \end{pmatrix} \tilde{Y}_2$$

and obtain

$$Y_1^{-1} A B^* Y_1 = \left(\begin{array}{c|cc} J & 0 & 0 \\ \hline 0 & N & * \\ 0 & 0 & 0 \end{array}\right), \; Y_2 B^* A Y_2^{-1} = \left(\begin{array}{c|cc} J & 0 & 0 \\ \hline 0 & N & 0 \\ 0 & * & 0 \end{array}\right). \tag{2.5}$$

From (2.5) we see, that AB^* and B^*A have the same Jordan blocks with respect to the nonzero eigenvalues. If σ is the smallest integer such that $N^\sigma = 0$ then

$$\begin{pmatrix} N & 0 \\ * & 0 \end{pmatrix}^{\sigma+1} = 0, \; \begin{pmatrix} N & * \\ 0 & 0 \end{pmatrix}^{\sigma+1} = 0,$$

which implies assertion (i).
Assertion (ii) follows from (2.2) in the case, when A has full rank. $\square$

As a direct consequence of this lemma we get

Corollary 2.6 *Let $A, S \in \mathrm{GL}(n, \mathbb{F})$, $A = S - FG$, where $F, G^T \in \mathrm{M}(n \times r, \mathbb{F})$, $r \leqslant n$. Set $S_c := I - GS^{-1}F$.*

(i) There exist $Y_1, Y_2 \in \mathrm{GL}(n, \mathbb{C})$ such that

$$Y_1^{-1} S^{-1} A Y_1 = \begin{pmatrix} J & 0 \\ 0 & J_1 \end{pmatrix}, \quad Y_2^{-1} S_c Y_2 = \begin{pmatrix} J & 0 \\ 0 & J_2 \end{pmatrix},$$

are both in Jordan canonical form and J_1 and J_2 have only ones on the main diagonal. If s, t are the smallest numbers such that $(J_1 - I)^s = 0, (J_2 - I)^t = 0$, then $|s - t| \leqslant 1$.

(ii) If F has full rank, then there exists $Y \in \mathrm{GL}(n, \mathbb{C})$ such that

$$Y^{-1}(AS^{-1})Y = \begin{pmatrix} S_c & * \\ 0 & I \end{pmatrix},$$

where the first r columns of Y are those of F.

Remark: This corollary shows, that the spectra of AS^{-1} and S_c are almost the same. Moreover they have almost the same Jordan canonical form. If also F has full rank, then S_c is the restriction of AS^{-1} to the invariant subspace spanned by the columns of F and several structures are inherited by S_c. E.g. we have a one to one correspondence between eigenvectors x of S_c and eigenvectors Fx of AS^{-1}.

One reason for the use of low rank modifications is their applicability in Krylov–subspace based methods as cg–like or semi–iterative methods [37]. For these methods the degree of the minimum polynomial of a matrix [3],p.47ff gives a theoretical upper bound for the number of iteration steps (See e.g. [3], pp.517–518). From Corollary 2.6 it follows that the minimum polynomial of $S^{-1}A$ is essentially given by the minimum polynomial of S_c. By Corollary 2.6 we have an interpretation of our coupling system S_c from (1.4). After at least one step of the iteration with $I - S^{-1}A$, i.e. for a given residual $r = b - Ax$ the approximate solution x is replaced $x - S^{-1}r$, the new residual is in a suitable small subspace, which ensures that cg–like and semi–iterative methods in exact arithmetic terminate after at most rank $W + 1$ steps, where $W = FX^{-1}G$.

2.2 Properties of the Coupling System in the Symmetric Positive Definite Case

For the class of symmetric, positive definite matrices we will now examine, how this property can be inherited by the coupling system. Naturally this is closely connected to the factorization of $W = FG$ in the splitting $A = S - W$. Even if A, S are symmetric, the factorization $W = FG$ need not to be of the form $W = FF^*$, since in general W is not necessarily positive semidefinite. If for example, S is a block diagonal part of A, then typically W is indefinite. However we will show that under some full rank assumptions for F, G the coupling system is still positive definite and self adjoint with respect to a suitably chosen inner product.
Consider a symmetric (Hermitian) matrix A with respect to the standard inner product $(\bullet, \bullet)$, and split A as $A = S - W$, where S is symmetric (Hermitian). We are interested in the properties of the coupling system S_c obtained as in (1.4).

Lemma 2.7 *Let* $A, S \in \mathrm{GL}(n, \mathbb{F})$, $A = A^*, S = S^*$, $A = S - FG$, *where* $F, G^T \in \mathrm{M}(n \times r, \mathbb{F})$. $S_c := I - GS^{-1}F$.

(i) If S is positive definite and if F has full rank, then S_c is self-adjoint with respect to $(F^*S^{-1}F\bullet, \bullet)$.

(ii) If A is positive definite and if F has full rank, then S_c is self-adjoint with respect to $(F^*A^{-1}F\bullet, \bullet)$.

(iii) If both matrices, S and A are positive definite and if F has full rank, then S_c is positive definite.

Proof:
From Corollary 2.6 we know that S_c has essentially the same spectrum as $S^{-1}A$ and AS^{-1}. If A and S are positive definite, then AS^{-1} is similar to $S^{-1/2}AS^{-1/2}$ and S_c must already have positive eigenvalues. So the third statement is a consequence of (i) and (ii).
We note, that $FS_c = AS^{-1}F$ and $S_c^{-1}G = GA^{-1}S$.

$$(F^*S^{-1}FS_cv, w) \quad = \quad (F^*S^{-1}AS^{-1}Fv, w)$$

$$\begin{aligned} &= (S^{-1}Fv, AS^{-1}Fw) \\ &= (S^{-1}Fv, FS_c w) \\ &= (F^*S^{-1}Fv, S_c w). \end{aligned}$$

So S_c is self-adjoint with respect to $(F^*S^{-1}F\bullet, \bullet)$ which implies the first statement. The proof of assertion 2 is analogous. □

Another way to preserve the positive definiteness consists in modifying the given initial splitting $A = S - W$ to $A = \hat{S} - \hat{W}$ such that in addition W is positive semidefinite. In principle this is always possible since $\hat{S} = A + \hat{W}$, i.e., we have to add a suitable positive semidefinite matrix $\hat{W}$ to A in order to obtain $\hat{S}$, which is then symmetric positive definite likewise. For $\hat{W}$ one has to perform a symmetric factorization of the form $\hat{W} = FX^{-1}F^*$. The corresponding coupling system $X - F^*\hat{S}^{-1}F$ is obviously symmetric. So the main question will be, when will the coupling system be also positive definite. We will show this under more general assumptions, which do not require the positive definiteness of A, S.

Lemma 2.8 *Let $A, S \in \mathrm{GL}\,(n, \mathbb{F})$, $A = A^*, S = S^*$, $A = S - FX^{-1}F^*$, where $F \in \mathrm{M}\,(n \times r, \mathbb{F})$, $X \in \mathrm{GL}\,(r, \mathbb{F}), X = X^*$. $S_c := X - F^*S^{-1}F$. Then*

$$\begin{pmatrix} S & \mathrm{O} \\ \mathrm{O} & S_c \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} A & \mathrm{O} \\ \mathrm{O} & X \end{pmatrix}$$

have the same inertia.

Proof:
Using the symmetric Schur–complement of the matrix

$$M = \begin{pmatrix} S & F \\ F^* & X \end{pmatrix}$$

we obtain

$$M = \begin{pmatrix} I & \mathrm{O} \\ F^*S^{-1} & I \end{pmatrix} \begin{pmatrix} S & \mathrm{O} \\ \mathrm{O} & X - F^*S^{-1}F \end{pmatrix} \begin{pmatrix} I & S^{-1}F \\ \mathrm{O} & I \end{pmatrix}$$

and

$$M = \begin{pmatrix} I & FX^{-1} \\ \mathrm{O} & I \end{pmatrix} \begin{pmatrix} S - FX^{-1}F^* & \mathrm{O} \\ \mathrm{O} & X \end{pmatrix} \begin{pmatrix} I & \mathrm{O} \\ X^{-1}F^* & I \end{pmatrix}.$$

As a direct consequence of Sylvester's law of inertia [41], p.416, we get that

$$\begin{pmatrix} S & \mathrm{O} \\ \mathrm{O} & X - F^*S^{-1}F \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} S - FX^{-1}F^* & \mathrm{O} \\ \mathrm{O} & X \end{pmatrix}$$

have the same inertia. □

Corollary 2.9 *Let $A, S \in \mathrm{GL}\,(n, \mathbb{F})$, $A = A^*, S = S^*$, $A = S - \sigma FF^*$, where $F \in \mathrm{M}\,(n \times r, \mathbb{F})$, $\sigma \in \mathbb{R}$ and set $S_c := I - \sigma F^*S^{-1}F$.*
If A and S have the same inertia, then S_c is positive definite.

This corollary simplifies the proof of Theorem 1 in [59].

As direct consequence of Corollary 2.6 and Lemma 2.8 we see that a linear system with S_c is at least as well conditioned as a linear system with $S^{-1}A$. More precisely, both systems have (nearly) the same distribution of eigenvalues. This is an important fact for preconditioned conjugate gradient methods.

Example 2.10 *Consider following matrix T and a block diagonal matrix S:*

$$T = \begin{pmatrix} 1 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{pmatrix}, S = \left(\begin{array}{cc|cc} 1 & -1 & & \\ -1 & 2 & & \\ \hline & & 2 & -1 \\ & & -1 & 2 \end{array}\right) \Rightarrow W \equiv S - A = \left(\begin{array}{cc|cc} 0 & 0 & & \\ 0 & 0 & 1 & \\ \hline & 1 & 0 & 0 \\ & & 0 & 0 \end{array}\right).$$

In order to apply Corollary 2.9 we have to modify W such that the modified W becomes either positive semidefinite or negative semidefinite. One choice could be

$$W := \left(\begin{array}{cc|cc} 0 & 0 & & \\ 0 & 1 & 1 & \\ \hline & 1 & 1 & 0 \\ & & 0 & 0 \end{array}\right) \Longrightarrow S = \left(\begin{array}{cc|cc} 1 & -1 & & \\ -1 & 3 & & \\ \hline & & 3 & -1 \\ & & -1 & 2 \end{array}\right).$$

The modified S will be nonsingular since we have added something nonnegative to the diagonal entries. In principle one could also make W negative semidefinite by inserting -1 instead of 1. But in this case the modified S will become singular.

2.3 Properties of S_c in the M–Matrix Case

Similar to the symmetric positive definite case, we will examine for the case of M–matrices which properties have to be required for the splitting $A = S - W$ and the factorization $W = FG$ in order to preserve the M–matrix property for the coupling system.

Definition 2.11 *$A \in \mathrm{GL}(n, \mathbb{R})$ is said to be inverse nonnegative, if its inverse is element-wise nonnegative.*
A is said to be an M–Matrix, if it is inverse nonnegative and its off–diagonal elements are nonpositive.

One important equivalent criterion for M–Matrices is given by the following lemma:

Lemma 2.12 *Let $A \in \mathrm{GL}(n, \mathbb{R})$ such that $a_{ij} \leqslant 0$ for any $i \neq j$, $i, j = 1, \ldots, n$. Then A is an M–Matrix if and only if there exists $c \in \mathbb{R}^n$, $c \succ 0$ such that $Ac \succ 0$.*

Proof:
See [8], pp.132ff □

Analogous to the positive definite case, where we required a positive semidefinite W, here we will require an element-wise nonnegative W.

Lemma 2.13 *Let $A, S \in \mathrm{GL}\,(n,\mathbb{R})$ where A is inverse nonnegative, $A = S - FG$, where $F, G^T \in \mathrm{M}\,(n \times r, \mathbb{R})$. $S_c := I - GS^{-1}F$. Assume that both matrices F and G are nonnegative or nonpositive. Then*

(i) S_c is inverse nonnegative.

(ii) If in addition, the off–diagonal entries of S are nonpositive, then A, S and S_c are M–Matrices.

Proof:
We note that

$$S_c^{-1} = I + GA^{-1}F, \tag{2.14}$$

which can be verified by straightforward computation. From this and the assumptions, (i) follows immediately.
For (ii) we note that if the off–diagonal of S are nonpositive then this also holds for the off–diagonal entries of A. Thus A is an M–Matrix. By Lemma 2.12 we obtain that $S = A + FG$ is an M–Matrix. Finally it follows that the off–diagonal entries of S_c have to be nonpositive, since $S^{-1} \succeq \mathrm{O}$. This completes the proof. $\square$

The assumptions of Lemma 2.13 imply that $W = FG \succeq \mathrm{O}$. If we obtain $W \succ \mathrm{O}$ from the splitting $A = S - W$, then there exists an obvious way to factorize W, if W is sparse. Let $i_1, \dots, i_r$ denote the nonzero rows of W, then W can be factorized as $W = [e_{i_1}, \dots, e_{i_r}]\,\tilde{W}$, where e_i denotes the i–th unit vector and $\tilde{W}$ denotes the nonzero rows of W.
In general, S_c need not be an M–Matrix if F or G do not satisfy the sign condition of the lemma.

Example 2.15 *Consider the matrix*

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix},$$

where all blocks of A have size $m \times m$

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}$$

This matrix arises from the so–called five point star discretization of Poisson's differential equation on a rectangle (see [49]). We split A similar to a block Jacobi like splitting: For suitable $m_1, \ldots, m_{p-1} \in \{1, \ldots, m-1\}$:

$$S = \begin{pmatrix} S_1 & & \\ & \ddots & \\ & & S_p \end{pmatrix} \tag{2.15}$$

and for $j = 1, \ldots, p$ the diagonal blocks of S are:

$$S_1 = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & \boxed{T-I} \end{pmatrix}, \tag{2.16}$$

$j = 2, \ldots, p-1$:

$$S_j = \begin{pmatrix} \boxed{T-I} & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & \boxed{T-I} \end{pmatrix}, \tag{2.17}$$

$$S_p = \begin{pmatrix} \boxed{T-I} & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix} \tag{2.18}$$

and $W = S - A$ equals

$$W = \left(\begin{array}{ccc|ccc|ccc} 0 & & & & & & & & \\ & \ddots & & & & & & & \\ & & -I & I & & & & & \\ \hline & & I & -I & & & & & \\ & & & & \ddots & & & & \\ & & & & & -I & I & & \\ \hline & & & & & I & -I & & \\ & & & & & & & \ddots & \\ & & & & & & & & 0 \end{array}\right).$$

W has many columns and rows, which are zero. Thus $W = V\tilde{W}V^T$ for a suitable block

diagonal matrix $\tilde{W}$:

$$\underbrace{\begin{pmatrix} \begin{array}{c|cc|c} I & & & \\ \hline & I & & \\ & & I & \\ \hline & & & I \end{array} \end{pmatrix}}_{V} \underbrace{\begin{pmatrix} \boxed{\begin{array}{c|c} -I & I \\ \hline I & -I \end{array}} & & \\ & \ddots & \\ & & \boxed{\begin{array}{c|c} -I & I \\ \hline I & -I \end{array}} \end{pmatrix}}_{\tilde{W}} \underbrace{\begin{pmatrix} \begin{array}{c|cc|c} I & & & \\ \hline & I & & \\ & & I & \\ \hline & & & I \end{array} \end{pmatrix}}_{V^T}.$$

We can write $\tilde{W}$ as $\tilde{W} = -\tilde{F}\tilde{F}^T$, where

$$\tilde{F} = \begin{pmatrix} -I & & \\ I & & \\ & \ddots & \\ & & -I \\ & & I \end{pmatrix}.$$

We choose $F = V\tilde{F}$ and $G^T = -V\tilde{F}$. Then $S_c = I - GS^{-1}F = I + \tilde{F}^T V^T S^{-1} V\tilde{F}$. We will show, that S_c has a block orientated sign pattern, i.e., the diagonal blocks of S_c are nonnegative and the off–diagonal blocks are nonpositive. We can derive this by a symbolic notation, where we set a $\oplus$ for an nonnegative block and a $\ominus$ for a nonpositive block in F and $V^*S^{-1}V$. We note that S^{-1} is nonnegative, because S is still an M–Matrix. Therefore $V^*S^{-1}V$ is a block diagonal matrix with nonnegative blocks:

$$\begin{pmatrix} \oplus & & & & & & & \\ & \oplus & \oplus & & & & & \\ & \oplus & \oplus & & & & & \\ & & & \oplus & \oplus & & & \\ & & & \oplus & \oplus & & & \\ & & & & & \oplus & \oplus & \\ & & & & & \oplus & \oplus & \\ & & & & & & & \oplus \end{pmatrix}.$$

Obviously $\tilde{F}$ has the following sign pattern:

$$\begin{pmatrix} \ominus & & & \\ \oplus & & & \\ & \ominus & & \\ & \oplus & & \\ & & \ominus & \\ & & \oplus & \\ & & & \ominus \\ & & & \oplus \end{pmatrix}.$$

Thus $\tilde{F}^T V^T S^{-1} V \tilde{F}$ has the following sign pattern:

$$\begin{pmatrix} \oplus & \ominus & & \\ \ominus & \oplus & \ominus & \\ & \ominus & \oplus & \ominus \\ & & \ominus & \oplus \end{pmatrix}.$$

As $S_c = I + \tilde{F}^T V^T S^{-1} V \tilde{F}$, this shows, that S_c has the suitable sign pattern for M–Matrices if and only if all T's are of order one. So in general, S_c can not be an M–Matrix, but it is still symmetric positive definite (Corollary 2.9).

Example 2.16 *Consider again the model problem:*

$$S = \begin{pmatrix} S_1 & & \\ & \ddots & \\ & & S_p \end{pmatrix}$$

and for $j = 1, \ldots, p$ choose the diagonal blocks of S as follows:

$$S_1 = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & \boxed{T+I} \end{pmatrix},$$

$j = 2, \ldots, p-1$:

$$S_j = \begin{pmatrix} \boxed{T+I} & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & \boxed{T+I} \end{pmatrix},$$

$$S_p = \begin{pmatrix} \boxed{T+I} & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix}.$$

In this case we have that $W = S - A$ has the form

$$W = \left(\begin{array}{ccc|ccc|ccc} 0 & & & & & & & & \\ & \ddots & & & & & & & \\ & & I & I & & & & & \\ \hline & & I & I & & & & & \\ & & & & \ddots & & & & \\ & & & & & I & I & & \\ \hline & & & & & I & I & & \\ & & & & & & & \ddots & \\ & & & & & & & & 0 \end{array}\right).$$

As above, W can be factored as $W = V\hat{W}V^T$.

$$\underbrace{\left(\begin{array}{c|c|c} I & & \\ \hline & I & \\ & & \\ & I & \\ \hline & & I \end{array}\right)}_{V} \underbrace{\begin{pmatrix} \boxed{\begin{array}{c|c} I & I \\ \hline I & I \end{array}} & & \\ & \ddots & \\ & & \boxed{\begin{array}{c|c} I & I \\ \hline I & I \end{array}} \end{pmatrix}}_{\hat{W}} \underbrace{\left(\begin{array}{c|ccc|c} I & & & & \\ \hline & I & & I & \\ \hline & & & & I \end{array}\right)}_{V^T}.$$

We can write $\hat{W}$ as $\hat{W} = \hat{F}\hat{F}^T$, where

$$\hat{F} = \begin{pmatrix} I & & \\ I & & \\ & \ddots & \\ & & I \\ & & I \end{pmatrix}.$$

Choosing $F = V\hat{F}$ and $G^T = V\hat{F}$ we obtain $S_c = I - GS^{-1}F = I - \hat{F}^T V^T S^{-1} V\hat{F}$. Applying Corollary 2.9 and Lemma 2.13 we get that S_c is symmetric, positive definite and an M–Matrix, i.e., S_c is a Stieltjes matrix.

Next we use a well–known result for inverse positive matrices and regular splittings (see [18], pp.118ff).

Definition 2.17 *Let $A \in \mathrm{GL}(n,\mathbb{R})$ be split as $A = S - W$. Such a splitting is called weak regular, if $S \in \mathrm{GL}\,(n,\mathbb{R})$ is inverse nonnegative and $S^{-1}W$ is nonnegative.*
The splitting is called regular, if $S \in \mathrm{GL}\,(n,\mathbb{R})$ is inverse positive and W is nonnegative.
For $z \in \mathbb{C}$, $r \geqslant 0$, we denote by $\mathrm{D}(z,r) := \{w \in \mathbb{C} : |z-w| < r\}$ the open disc with center z and radius r in the complex plane and by $\overline{\mathrm{D}}(z,r) := \{w \in \mathbb{C} : |z-w| \leqslant r\}$ its closure.
Denote by $\Lambda(A)$ the set of eigenvalues of A and by $\rho(A)$ the largest eigenvalue of A in modulus.

Lemma 2.18 *Let $A \in \mathrm{GL}(n,\mathbb{R})$ be inverse nonnegative and consider a weak regular splitting $A = S - W$, where $W = FG$, $F, G \in \mathrm{M}(n \times r,\mathbb{R})$. Set $S_c = I - Gs^{-1}F$. Then*

(i) $\rho(I - S^{-1}A) = \dfrac{\rho(A^{-1}W)}{1 + \rho(A^{-1}W)} < 1.$

(ii) $\Lambda(S_c) \subset \overline{\mathrm{D}}(1, \rho(I - S^{-1}A)) \subset \mathrm{D}(1,1).$

(iii) $\min\limits_{\lambda \in \Lambda(S_c)} |\lambda| = 1 - \rho(I - S^{-1}A), \quad \rho(S_c) \leqslant 1 + \rho(I - S^{-1}A).$

Proof:
For the first assertion see [18], p.119.
By Corollary 2.6 assertion (ii) is a direct consequences of the first one.
The third assertion follows from assertion (ii) and the well–known Perron–Frobenius theory [82] for nonnegative matrices. □

We also note a well–known result, see [49],p.158 for inverse nonnegative matrices and the relations between the spectral radii of two regular splittings:

Lemma 2.19 *Let $A \in \mathrm{GL}(n, \mathbb{R})$ be inverse nonnegative and consider two regular splittings, $A = S_1 - W_1 = S_2 - W_2$. Assume that $W_1 \preceq W_2$ (component-wise). Then*

$$\rho(I - S_1^{-1}A) \leqslant \rho(I - S_2^{-1}A) < 1.$$

If in addition A^{-1} is element-wise positive and $W_1 \prec W_2$, then

$$\rho(I - S_1^{-1}A) < \rho(I - S_2^{-1}A) < 1.$$

As a consequence of these two Lemmata, we see, that under the assumptions of Lemmata 2.18 and 2.19 with $W_1 = F_1G_1, W_2 = F_2G_2$ the related coupling systems $S_{c1} = I - G_1S_1^{-1}F_1, S_{c2} = I - G_2S_2^{-1}F_2$ have the property, that the eigenvalues of S_{c1} are enclosed in a smaller disc. This means, the smaller the matrix W in the splitting $A = S - W$, the smaller the disc is in which the eigenvalues are included.

2.4 Properties of S_c in the Symmetric M–Matrix Case

At last we consider the case of symmetric positive definite M–Matrices, i.e. Stieltjes–Matrices, and discuss the question whether we are able to preserve the M–Matrix property and the positive definiteness for the coupling system at same time.

Lemma 2.20 *Let $A \in \mathrm{GL}\,(n, \mathbb{R})$ be a symmetric M–Matrix. Assume that A is split as $A = S - W$. Then the following are equivalent:*

(i) S is a symmetric M–Matrix and W symmetric and nonnegative;

(ii) $s_{ii} \geqslant a_{ii}$ for all i and $0 \geqslant s_{ij} = s_{ji} \geqslant a_{ij}$ for all $i \neq j$.

Proof:
(i) $\Longrightarrow$ (ii) is trivial.
If the inequalities of (ii) hold, then it follows immediately by element-wise comparison, that S and W have the correct sign pattern. Applying Lemma 2.12 we obtain that S is an M–Matrix. □

In the next theorem we will show that a splitting quite similar to the one in Lemma 2.20 can be modified by some changes in the diagonal entries to a splitting which gives in addition a coupling system which is also a symmetric M–Matrix.

Theorem 2.21 *Let $A \in \mathrm{GL}(n, \mathbb{R})$ be a symmetric M–Matrix. Assume that A is split as $A = S - W$ such that $s_{ii} = a_{ii}$ for all i and $0 \geqslant s_{ij} = s_{ji} \geqslant a_{ij}$ for all $i \neq j$. Define $\hat{S} := S + \operatorname{diag}(\sum_j w_{1j}, \ldots, \sum_j w_{nj})$. Then $A = \hat{S} - \hat{W}$ where $\hat{S}$ is a symmetric M–Matrix and $\hat{W}$ is nonnegative. In addition $\hat{W}$ can be factorized as*

$$\hat{W} = \sum_{\substack{i<j \\ w_{ij}>0}} \{\sqrt{w_{ij}}(e_i+e_j)\}\{\sqrt{w_{ij}}(e_i+e_j)\}^T = \underbrace{\left[\ldots, \sqrt{w_{ij}}(e_i + e_j), \ldots\right]}_{F} \underbrace{\left[\ldots, \sqrt{w_{ij}}(e_i + e_j), \ldots\right]^T}_{F^T},$$

where the $e_1, \ldots, e_n$ are the unit vectors and the matrix $F = \left[\ldots, \sqrt{w_{ij}}(e_i + e_j), \ldots\right]$ contains all columns $\sqrt{w_{ij}}(e_i + e_j)$ for $i < j, w_{ij} > 0$ in a suitable order. The coupling system $S_c = I - F^T \hat{S}^{-1} F$ is also a symmetric M–Matrix.

Proof:
It follows directly from Lemma 2.20 that $\hat{S}$ is a symmetric M–Matrix and that $\hat{W}$ is nonnegative. $\hat{W}$ can be disassembled as a sum of essential 2×2 matrices

$$\hat{W} = \sum_{i<j} [e_i, e_j] \begin{pmatrix} w_{ij} & w_{ij} \\ w_{ij} & w_{ij} \end{pmatrix} [e_i, e_j]^T .$$

By construction $\begin{pmatrix} w_{ij} & w_{ij} \\ w_{ij} & w_{ij} \end{pmatrix}$ has rank 1 and a factoriztion

$$\begin{pmatrix} w_{ij} & w_{ij} \\ w_{ij} & w_{ij} \end{pmatrix} = \sqrt{w_{ij}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \sqrt{w_{ij}}(1,\ 1).$$

The resulting factorization has all properties to ensure that S_c is both, M–Matrix and symmetric positive definite. $\square$

Remark:

- It follows immediately from Theorem 2.21 that the order of S_c is equal to the number of nonzero entries in $\hat{W}$ above the diagonal, while the rank of F is less than or equal to the number of nonzero rows/columns in $\hat{W}$. So in general one can not expect that F has full rank.

- By Lemma 2.18 we know that the eigenvalues of $S^{-1}A$ and $\hat{S}^{-1}A$ lie in the interval $(0, 2)$. According to Lemma 2.19 the consequence in the change of the diagonal entries is that the largest eigenvalue of $\hat{S}^{-1}A$ is now less than or equal to 1, because $\hat{W}$ is positive semidefinite. The smallest eigenvalue of the $\hat{S}^{-1}A$ is less than or equal to the smallest eigenvalue of $S^{-1}A$. This may not be a disadvantage for the use of $\hat{S}$

as preconditioner to A in the cg–method. It has been shown in [4],[5] that if the spectrum of $\hat{S}^{-1}A$ satisfies

$$\Lambda(\hat{S}^{-1}A) \subseteq [a,b] \cup \{1\},$$

where $0 < a \leqslant b < 1$, then the number of iterations of the cg–method needed to obtain $\|x - x^{(k)}\|_{A^{1/2}} \leqslant \varepsilon \|x - x^{(0)}\|_{A^{1/2}}$ is at most

$$k = \lceil \frac{1}{2}\sqrt{\frac{b}{a}} \ln \frac{2}{\varepsilon} + 2 \rceil .$$

This gives a better bound than the conventional estimate [41], p.525

$$k = \lceil \frac{1}{2}\sqrt{\frac{1}{a}} \ln \frac{2}{\varepsilon} + 1 \rceil = \lceil \frac{1}{2}\sqrt{\operatorname{cond}_2(\hat{S}^{-1}A)} \ln \frac{2}{\varepsilon} + 1 \rceil .$$

In other words: Instead of the condition number $\frac{1}{a}$ the ratio $\frac{b}{a}$ of the largest eigenvalue b less than 1 and the smallest eigenvalue a determines the convergence speed of the cg–method.

Summary

In this chapter we have shown that the coupling system S_c can be interpreted as the restriction of the preconditioned matrix AS^{-1} to a special invariant subspace. Several algebraic properties can be obtained from this fact. For two special classes of matrices we have also discussed how the structures of the initial system can be inherited by the coupling system.

From our list of questions on Page 10 we have partially given an answer to the first and the second question.
Solving a linear system with the matrix S_c can still be difficult, if the coupling system is ill–conditioned even if the coupling systems inherits structures of the initial system. For Schur–complement methods a better conditioned system is constructed by the use of suitable preconditioners. Typically the construction of preconditioners requires either information about the underlying problem or an explicit representation of the given matrix. For the coupling system from the Divide & Conquer approach an ill–conditioned system can be viewed as consequence of an unsatisfactory splitting. This means, that instead of constructing preconditioners for the coupling system we can construct a more suitable splitting $A = S - W$ in order to improve the properties of the coupling system. The question is, how we have to modify a given splitting $A = S - W$ to obtain a more suitable splitting $A = \hat{S} - \hat{W}$. This will be the topic of the next chapter.

Chapter 3

Nested Divide & Conquer

In this chapter we will discuss the nested use of the Sherman–Morrison–Woodbury formula (1.4) for divide & conquer methods. The reason for the nested use is, that the coupling system arising from the splitting $A = S - FG$ by the divide & conquer approach can be ill–conditioned or for practical purposes the order of the coupling system may still be too large. The nested use of the Sherman–Morrison–Woodbury formula deals with both problems.

3.1 Motivation

We will first give a motivation on the nested use of the Sherman–Morrison–Woodbury formula. The idea is to adaptively construct a nested sequence of splittings $S = S_0, S_1, \ldots, S_m$ where the rank of the remaining matrix is reduced at each step. As motivation we will only consider one step of this nested construction. Assume that we have a splitting

$$A = S - FG \equiv S_0 - F_0 G_0,$$

where $A, S_0 \in \mathrm{GL}\,(n, \mathbb{F})$, $F_0, G_0^T \in \mathrm{M}\,(n \times r, \mathbb{F})$. Now it may happen that S_0 is ill–conditioned or even worse S_0 may be well–conditioned but the corresponding coupling system $S_{c,0} = I - G_0 S_0^{-1} F_0$ may be ill–conditioned. On one hand, if we would like to apply a method of Krylov–subspace type [37] to $S_{c,0}$, this may lead to slow convergence or even no convergence at all. On the other hand a direct solution of $S_{c,0}$ is sensitive to perturbations. Above all this is advisable only if the size r of $S_{c,0}$ is small. So it may be useful to replace S_0 by a more suitable approximation, which can still be easily solved on a parallel machine. A natural choice can be $S_1 = S_0 - \tilde{F}_0 \tilde{G}_0$, where $\tilde{F}_0, \tilde{G}_0^T \in \mathrm{M}\,(n \times s, \mathbb{F})$, $s \ll r$, e.g. $s = 1$. Solving a system with S_1 is almost as easy as with S_0 if we just apply the Sherman–Morrison–Woodbury formula to S_1. Once we have computed the resulting small $s \times s$ coupling system $\tilde{S}_{c,0} = I - \tilde{G}_0 S_0^{-1} \tilde{F}_0$, we can solve this small system directly.

$$S_1^{-1} = S_0^{-1} + S_0^{-1} \tilde{F}_0 \tilde{S}_{c,0}^{-1} \tilde{G}_0 S_0^{-1}$$

We will illustrate this by an example.

Example 3.1 *Let $n = 30$ and consider matrices A and S, where*

$$A \equiv A_n = \begin{pmatrix} 3 & -2 & & & 0 \\ -1 & 3 & -2 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 3 & -2 \\ 0 & & & -1 & 3 \end{pmatrix} \in \mathbb{R}^{n,n},\ S = \left(\begin{array}{c|c|c} A_{n/3} & & O \\ \hline & A_{n/3} & \\ \hline O & & A_{n/3} \end{array}\right).$$

For simplicity let us factorize the remaining matrix $W = S - A$ as $W = FG$, where

$$F = \left(\begin{array}{c|c} O & \\ 1\ \ 0 & \\ \hline 0\ \ 1 & \\ O & O \\ & 1\ \ 0 \\ \hline & 0\ \ 1 \\ & O \end{array}\right) \in \mathbb{R}^{n,4},\ G = \left(\begin{array}{cc|cc|cc|cc} O & 0 & 2 & O & & & & \\ & 1 & 0 & & & & & \\ \hline & & & O & 0 & 2 & O & \\ & & & & 1 & 0 & & \end{array}\right) \in \mathbb{R}^{4,n}.$$

Using MATLAB [60] we find that the condition number of A is very large, $\operatorname{cond}_2(A) \approx 3.9 \cdot 10^{10}$ (for arbitrary n it will be $\approx 2^n$). The condition number of S will be approximately $\operatorname{cond}_2(S) \approx 2 \cdot 10^4$ which is much better than the condition number of A. But the condition number of S_c is worse, $\operatorname{cond}_2(S_c) \approx 1.2 \cdot 10^7$. By Theorem 2.6 the matrix AS^{-1} is closely connected to S_c, $\operatorname{cond}_2(AS^{-1}) \approx 5.7 \cdot 10^7$ which is of the same magnitude as the condition number of S_c. Consider now the following two choices of rank 1 updates.

$$\tilde{F} = F \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T,\ \tilde{G} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} G,$$

$$\bar{F} = F \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix}^T,\ \bar{G} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix} G.$$

We set $\tilde{S} = S - \tilde{F}\tilde{G}$, $\bar{S} = S - \bar{F}\bar{G}$. In order to solve one system with $\tilde{S}$ or $\bar{S}$ we can apply the Sherman–Morrison formula (1.4). To do this we have to solve systems with $\tilde{F}, \bar{F}$ as right hand sides, i.e.

$$\tilde{E} = S^{-1}\tilde{F},\ \bar{E} = S^{-1}\bar{F}.$$

By setting $\tilde{S}_c = 1 - \tilde{G}\tilde{E}$, $\bar{S}_c = 1 - \bar{G}\bar{E}$ we obtain

$$\tilde{S}^{-1} = (I + \tilde{E}\frac{1}{\tilde{S}_c}\tilde{G})S^{-1},\ \bar{S}^{-1} = (I + \bar{E}\frac{1}{\bar{S}_c}\bar{G})S^{-1}.$$

It turns out that solving a system with $\tilde{S}, \bar{S}$ is only slightly more expensive than solving a system with S, after $\tilde{E}, \bar{E}$ have been computed. The interesting question is, whether we did improve the properties of $\tilde{S}^{-1}, A\tilde{S}^{-1}$, $\bar{S}^{-1}, A\bar{S}^{-1}$ by this rank 1 update or not. One can verify that

$\operatorname{cond}_2(\tilde{S}^{-1})$	$\operatorname{cond}_2(A\tilde{S}^{-1})$	$\operatorname{cond}_2(\bar{S}^{-1})$	$\operatorname{cond}_2(A\bar{S}^{-1})$
$2.2 \cdot 10^4$	$4.8 \cdot 10^7$	$3.1 \cdot 10^7$	$9.4 \cdot 10^4$

We see that the condition number of $\tilde{S}$ is almost the same as that of S, while the condition number of $\bar{S}$ is closer to that of A. From this point of view $\tilde{S}$ seems to be a better choice than $\bar{S}$, but anyway we only have to solve a system with S while the remaining part is done by a rank 1 update. So the increase in the condition number of $\bar{S}$ is not so critical. From the practical point of view it is much more interesting how the preconditioned systems $A\tilde{S}^{-1}$ and $A\bar{S}^{-1}$ behave. Here the condition number of $A\tilde{S}^{-1}$ does not essentially differ from that of AS^{-1} while the condition number of $A\bar{S}^{-1}$ has been extremely reduced.

Example 3.1 shows that when replacing the initial matrix S_0 by a low rank modification $S_1 = S_0 - \tilde{F}_0\tilde{G}_0$ only slightly more work is necessary to solve a system with S_1 than to solve a system with S_0. In this case we obtain

$$A = S_0 - F_0G_0 = S_1 - F_1G_1,$$

where $S_1 = S_0 - \tilde{F}_0\tilde{G}_0$ and $\tilde{F}_0, \tilde{G}_0$ have small rank.
The question is, in which way we should choose $\tilde{F}_0, \tilde{G}_0$. A natural criterion will be, that the resulting system splitting $A = S_1 - F_1G_1$, where $F_1, G_1^T \in \mathrm{M}(n \times (r-s), \mathbb{F})$ should have a smaller rank if we replace S_0 by S_1. By again using the Sherman–Morrison–Woodbury formula (1.4) for $A = S_1 - F_1G_1$ we obtain

$$A^{-1} = S_1^{-1} + S_1^{-1}F_1S_{c,1}^{-1}G_1S_1^{-1}, \text{ with } S_{c,1} = I - G_1S_1^{-1}F_1$$

and the size of $S_{c,1}$ has just been reduced by s compared with the initial coupling system $S_c \equiv S_{c,0}$.

Example 3.2 *In Example 3.1 our choices of $\tilde{F}, \tilde{G}$ and $\bar{F}, \bar{G}$ will have a resulting splitting $A = S_1 - F_1G_1$ with matrices F_1G_1 of rank 3. For $\tilde{F}, \tilde{G}$ this is trivial by taking the last three rows/columns of F, G as F_1, G_1 and for $\bar{F}, \bar{G}$ use*

$$F_1 = F \begin{pmatrix} 1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad G_1 = \begin{pmatrix} 1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^T G.$$

Another criterion for the choice of $\tilde{F}_0, \tilde{G}_0$ should be of course, that at least the properties of the new coupling system $S_{c,1} = I - G_1S_1^{-1}F_1$ should be improved, e.g. it should have a better condition number than the previous one.

Example 3.3 *In Example 3.1 the initial coupling system $I - GS^{-1}F$ will be*

$$S_{c,0} = \begin{pmatrix} 1 & -9.9951 \cdot 10^{-1} & -5.0024 \cdot 10^{-1} & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -9.9951 \cdot 10^{-1} \\ 0 & -4.8852 \cdot 10^{-4} & -4.9976 \cdot 10^{-1} & 1 \end{pmatrix}.$$

As mentioned earlier, its condition number is $\mathrm{cond}_2(S_{c,0}) \approx 1.2 \cdot 10^7$. *For our choice of* $\tilde{F}, \tilde{G}$ *we get for* $S_1 = S_0 - \tilde{F}\tilde{G}$ *a small coupling system* $\tilde{S}_{c,0} = 1 - \tilde{G}S^{-1}\tilde{F} = 1$. *For* $A = S_1 - F_1G_1$ *a new coupling system* $S_{c,1} = I - G_1S_1^{-1}F_1$ *of size* 3×3 *is obtained,*

$$S_{c,1} = \begin{pmatrix} 4.8852 \cdot 10^{-4} & -5.0024 \cdot 10^{-4} & 0 \\ 0 & 1 & -9.9951 \cdot 10^{-1} \\ -4.8852 \cdot 10^{-4} & -4.9976 \cdot 10^{-1} & 1 \end{pmatrix}.$$

One can easily verify that its condition number is $6.5647 \cdot 10^6$ *while for the other choice* $\bar{F}, \bar{G}$ *the condition number of the corresponding* $S_{c,1}$ *will be* $8.1870 \cdot 10^3$.

It is clear that we can successively repeat the strategy of replacing S_0 by S_1 leading to a nested sequence of splittings $S_0, S_1, \ldots, S_m$ and therefore we obtain a nested sequence of coupling systems $S_{c,0}, S_{c,1}, \ldots, S_{c,m}$.

We will now discuss the following topics.

- A formal description of the nested divide & conquer process as motivated above.
- The properties of the sequence of coupling systems.
- The choice of low rank modifications.

3.2 General Construction of Nested Splittings

We start with some general comments on the construction of nested splittings.

We will briefly describe the idea behind the construction of a nested sequence of splittings. Assume that an initial splitting $A = S_0 - F_0G_0$ is given. In order to construct a low rank splitting of the form $S_1 = S_0 - \tilde{F}_0\tilde{G}_0$ and a resulting splitting $A = S_1 - F_1G_1$ we introduce a matrix $\tilde{U} \in \mathrm{M}(r \times s, \mathbb{F})$ such that $\tilde{U}^*\tilde{U} = I_s$ and extend it to an orthogonal matrix $U = [\tilde{U}, \hat{U}]$. Then we set $\tilde{F}_0 = F_0\tilde{U}$, $\tilde{G}_0 = \tilde{U}_0^*G_0$. Since $UU^* = I$ and $F_0G_0 = \tilde{F}_0\tilde{G}_0 + F_0\hat{U}\hat{U}^*G_0$ we obtain in a natural way $F_1 = F_0\hat{U}$, $G_1 = \hat{U}_1^*G_0$ and thus S_1.
This strategy can be obviously generalized in the following way. Consider two matrices $\tilde{U}, \tilde{V}^T \in \mathrm{M}(r \times s, \mathbb{F})$ such that $\tilde{V}\tilde{U} = I$, i.e., the columns of $\tilde{U}$ and the rows of $\tilde{V}$ are biorthogonal. In this case both matrices can be extended to matrices $U = [\tilde{U}, \hat{U}]$, $V = [\begin{smallmatrix}\tilde{V}\\ \hat{V}\end{smallmatrix}]$ such that $VU = I$.
Here we will restrict ourselves to the use of orthogonal matrices U. The more general case has already been discussed in [10].

The general construction can be described as follows. Given an initial splitting $A = S - FG \equiv S_0 - F_0G_0$ we consider numbers $s_0, s_1, \ldots, s_{m-1}$ with $\sum_{i=0}^{m-1} s_i < r$ and define successively $r_0 = r$, $r_{k+1} = r_k - s_k$, $k = 0, \ldots, m-1$. We consider a sequences of orthogonal matrices $U_k \in \mathrm{GL}\,(r_k, \mathbb{F})$ partitioned as $U_k = [\tilde{U}_k, \hat{U}_k]$, where $\tilde{U}_k$ has size $r_k \times s_k$ and $\hat{U}_k$ has size $r_k \times r_{k+1}$. Using this sequence of matrices we will define a nested sequence of splittings.

$$S_0 := S,\ F_0 := F,\ G_0 := G \tag{3.4}$$

$$(3.5)\qquad k=0,\dots,m-1:\quad \left\{\begin{array}{rcl} \left[\tilde F_k,\, F_{k+1}\right] & := & \left[F_k\tilde U_k,\, F_k\hat U_k\right] = F_kU_k \\ \left[\begin{array}{c}\tilde G_k\\ G_{k+1}\end{array}\right] & := & \left[\begin{array}{c}\tilde U_k^*G_k\\ \hat U_k^*G_k\end{array}\right] = U_k^*G_k \\ S_{k+1} & := & S_k-\tilde F_k\tilde G_k \end{array}\right.$$

By the definition of the nested sequence of splittings we are now in the situation that

$$(3.6)\qquad S_{k+1}=S_k-\tilde F_k\tilde G_k,\ k=0,\dots,m-1,\ \ A=S_k-F_kG_k,\ k=0,\dots,m.$$

The sequence of nested splittings as it has been defined so far involves two sequences of coupling systems. One sequence of small coupling systems $\tilde S_{c,k}$ from the relation $S_{k+1}=S_k-\tilde F_k\tilde G_k$ and another sequence of resulting coupling systems $S_{c,k}$ from the relation $A=S_k-F_kG_k$. They will be defined by

$$(3.7)\qquad \begin{array}{rcl} \tilde S_{c,k} & := & I-\tilde G_kS_k^{-1}\tilde F_k,\ k=0,\dots,m-1,\\ S_{c,k} & := & I-G_kS_k^{-1}F_k,\ k=0,\dots,m. \end{array}$$

Using twice the Sherman–Morrison–Woodbury formula (1.4) we obtain

$$(3.8)\qquad \begin{array}{rcl} S_{k+1}^{-1} & = & S_k^{-1}+S_k^{-1}\tilde F_k\tilde S_{c,k}^{-1}\tilde G_kS_k^{-1},\ k=0,\dots,m-1\\ A^{-1} & = & S_k^{-1}+S_k^{-1}F_kS_{c,k}^{-1}G_kS_k^{-1},\ k=0,\dots,m \end{array}$$

The successive use of nested splittings generates in every step a remaining coupling system $S_{c,k}$. Even if the reduced coupling system $S_{c,k}$ is not better conditioned than $S_{c,k-1}$, at least the size has been reduced by s_k.

So in any step of the successively defined splittings we have the following coupling systems.

k	splitting	small coupling systems	remaining coupling system
0	$A=S-FG$		S_c
1	$A=S_1-F_1G_1$	$\tilde S_{c,0}$	$S_{c,1}$
2	$A=S_2-F_2G_2$	$\tilde S_{c,0},\tilde S_{c,1}$	$S_{c,2}$
		$\vdots$	
m	$A=S_m-F_mG_m$	$\tilde S_{c,0},\dots,\tilde S_{c,m-1}$	$S_{c,m}$

Before we will give an algorithmic description of the generation and the way of applying the nested sequence splittings, we will briefly comment on the equivalent approach using left inverses in Section 1.3. In (1.7) we have assumed that rank $F=r$ and that $H\in \mathrm{M}\,(n\times r,\mathbb{F})$ is chosen such that $HF=I$. This assumption has lead to an equivalent way to write A^{-1} as it is shown in (1.9).

$$A^{-1}=S^{-1}+S^{-1}FT_c^{-1}H(I-AS^{-1}),\ \text{where } T_c=HAS^{-1}F.$$

We easily get left inverse matrices $\tilde{H}_k, H_{k+1}$ analogously to the construction of $\tilde{G}_k, G_{k+1}$.

$$H_0 = H, \begin{bmatrix} \tilde{H}_k \\ H_{k+1} \end{bmatrix} := \begin{bmatrix} \tilde{U}_k^* H_k \\ \hat{U}_k^* H_k \end{bmatrix} = U_k^* H_k, \; k = 0, \dots, m-1. \tag{3.9}$$

Analogously to (3.4)–(3.8) we can successively use the low rank modification formula (1.9). Since the nested generation for the left inverse approach is quite analogous to the first description in (3.4)–(3.8) we need not go into detail to describe the analogous procedure for the left inverse approach. This has been done in [10].

3.3 Algorithmic Description of Nested Divide & Conquer Methods

In this section we will give an abstract algorithm for the generation of nested splittings. From the definition of the nested sequence in (3.4)–(3.8) it seems as if the application of $A^{-1} = S_k^{-1} + S_k^{-1} F_k S_{c,k}^{-1} G_k S_k^{-1}$ with $S_{c,k} = I - G_k S_k^{-1} F_k$ involves a exponential call of $S_k, S_{k-1}, S_{k-2}, \dots$ when used for larger numbers k, since S_k^{-1} appears four times in the Sherman–Morrison formula. One can avoid this, if in any step of the generation of nested splittings a matrix $E_k = S_k^{-1} \tilde{F}_k$ is once computed and used in further steps. Using this E_k we immediately obtain

$$\tilde{S}_{c,k} = I - \tilde{G}_k E_k \tag{3.10}$$

and

$$S_{k+1}^{-1} = (I + E_k \tilde{S}_{c,k}^{-1} \tilde{G}_k) S_k^{-1} = (I + E_k \tilde{S}_{c,k}^{-1} \tilde{G}_k) \cdots (I + E_0 \tilde{S}_{c,0}^{-1} \tilde{G}_0) S^{-1}. \tag{3.11}$$

From this we can derive an algorithm for solving systems with S_k.

Algorithm 3.12 (Solving systems with nested D & C operators)
Let $A \in \mathrm{GL}(n, \mathbb{F})$ be split as $A = S - FG$, where $F, G^T \in \mathrm{M}(n \times r, \mathbb{F})$. Using the notation of (3.4)–(3.11), we assume that all $S_0, \dots, S_k$ are nonsingular. Suppose in addition that $E_0, \dots, E_{k-1}$ are already computed and that for any $l = 0, \dots, k-1$ an LU decomposition $\tilde{S}_{c,l} = L_{c,l} U_{c,l}$ has been computed. Then a system $S_k x = b$ can be solved in the following way:

```
Solve $Sx = b$
for $l = 0, 1, \dots, k-1$
    $\rho := \tilde{G}_l x$
    Solve $L_{c,l} U_{c,l} \delta = \rho$ directly
    $\gamma := E_l \delta$
    $x := x + \gamma$
```

For Algorithm 3.12 we have required that $E_0, \dots, E_{k-1}$ have already been computed and that an LU decomposition has been computed for $\tilde{S}_{c,l}$. To construct the nested sequence of splittings itself and to provide these assumptions we have to formulate another algorithm.

Before we do this we will briefly comment on one technical detail. Note that by (3.5), $\tilde{F}_k$ and $\tilde{G}_k$ can be directly obtained from F, G without using F_{k-1}, G_{k-1}:

(3.13) $$\tilde{F}_k = FY_k, \tilde{G}_k = Y_k^* G, \quad \text{where } Y_k = \hat{U}_0(\cdots \hat{U}_{k-2}(\hat{U}_{k-1}\tilde{U}_k)\cdots).$$

Analogously F_k, G_k satisfy

(3.14) $$F_k = F\hat{U}_0 \cdots \hat{U}_k, G_k = \hat{U}_k^* \cdots \hat{U}_0^* G.$$

Thus in order to compute $\tilde{F}_k, \tilde{G}_k$ we need not explicitly compute F_{k-1}, G_{k-1} but only Y_k and then apply it to F, G. This is important, since for small s_k, F_{k-1} and G_{k-1} will have almost r columns/rows and consequently their computation may be expensive, while Y_k only has s_k columns and the application to F, G will probably be cheap, since F, G are sparse.

Algorithm 3.15 (Generation of nested D & C operators)
Let $A \in \mathrm{GL}(n, \mathbb{F})$ be split as $A = S - FG$, where $F, G^T \in \mathrm{M}(n \times r, \mathbb{F})$. Using the notation of (3.4)–(3.11), we assume that all $S_0, \ldots, S_k$ are nonsingular.

```
$S_0 \equiv S$, $r_0 \equiv r$.
for $k = 0, \ldots, m-1$
    Consider $s_k \ll r_k$ and $\tilde{U}_k \in \mathrm{M}(r_k \times s_k, \mathbb{F})$ such that $\tilde{U}_k^* \tilde{U}_k = I$ and
    expand $\tilde{U}_k$ to an orthogonal matrix $U_k = \left[\tilde{U}_k, \hat{U}_k\right]$.
    $r_{k+1} := r_k - s_k$.
    $Y_k := \hat{U}_0(\cdots \hat{U}_{k-2}(\hat{U}_{k-1}\tilde{U}_k)\cdots)$
    $\tilde{F}_k := FY_k$
    Solve $S_k E_k = \tilde{F}_k$ applying Algorithm 3.12.
    $\tilde{S}_{c,k} := I - \tilde{G}_k E_k$.
    Factorize $\tilde{S}_{c,k} = L_{c,k} U_{c,k}$ (LU decomposition)
```

Algorithm 3.15 itself calls Algorithm 3.12. But at this stage $E_0, \ldots, E_{k-1}$ are already been computed. Also an LU decomposition has been performed for $\tilde{S}_{c,0}, \ldots, \tilde{S}_{c,k-1}$.

The interaction between Algorithm 3.15 and Algorithm 3.12 is shown in the following table.

k	to compute by Alg. 3.15	provided by Alg. 3.15 for Alg. 3.12	call of Alg. 3.12
0	E_0 $\tilde{S}_{c,0} = L_{c,0}U_{c,0}$	—	no
1	E_1 $\tilde{S}_{c,1} = L_{c,1}U_{c,1}$	E_0 $L_{c,0}U_{c,0}$	yes, with k=1
	$\vdots$		
m	E_m $\tilde{S}_{c,m} = L_{c,m}U_{c,m}$	$E_0, \ldots, E_{m-1}$ $L_{c,0}U_{c,0}, \ldots, L_{c,m-1}U_{c,m-1}$	yes, with k=m

Remark: By construction we have

$$A = S_m - F_m G_m. \tag{3.16}$$

This leads to a remaining coupling system

$$S_{c,m} \;:=\; I - G_m S_m^{-1} F_m = \hat{U}_{m-1}^* \cdots \hat{U}_0^* (I - G S_m^{-1} F) \hat{U}_0 \cdots \hat{U}_{m-1} \tag{3.17}$$

As long as we use a Krylov–subspace based method [41], pp.475ff, [37] for solving a system with $S_{c,m}$, we only have to multiply vectors with $F\hat{U}_0 \cdots \hat{U}_{m-1}$, $\hat{U}_{m-1}^* \cdots \hat{U}_0^* G$. Therefore we do not have to assemble the product explicitly. If we would have to form the product, this would be expensive, since we have assumed that each s_k is small compared with r. So if m is not too large, each $\hat{U}_l$ would be almost an $r \times r$ matrix, while in principle a multiplication with $\hat{U}_k$ can be done in $\mathcal{O}(s_k)$ steps because of the special relation to $\tilde{U}_k$.
It might be sensible to compute the product $GS_m^{-1}F$ explicitly, since this can be easily done in parallel. If we would solve the coupling system explicitly this would be even necessary. Beside this fact an explicit computation of $GS_m^{-1}F$ might be useful if multiplications with $S_{c,m}$ have to be performed several times to avoid the more expensive solution of systems with S_m.

The general construction of nested splittings and its algorithmic description involve several questions.
Conditions have to be formulated in order to guarantee the rigorous assumption that all S_k are nonsingular. The properties of the sequence of small coupling systems $\tilde{S}_{c,k}$ and remaining coupling systems have to be examined. Especially the properties of the remaining coupling $S_{c,m}$ are of great interest, since the nested use of splittings has been introduced to improve the properties of $S_{c,m}$. An answer to these problems will given in the next section. Finally one has to discuss the choice of $\tilde{U}_k$. This will be done lateron.

3.4 Algebraic Properties of Coupling Systems Obtained by Nested Divide & Conquer Methods

In this section we will discuss conditions which ensure that the sequence of nested splittings defined by (3.5),(3.6) will exist. We will see that this is closely related to the choice of $\tilde{U}_k$ in (3.5). In addition we may ask about the properties of the sequence of small coupling systems $\tilde{S}_{c,k}$ from (3.7) and especially the last remaining coupling system

$$S_{c,m} = \hat{V}_m \cdots \hat{V}_1 (I - G S_m^{-1} F) \hat{U}_1 \cdots \hat{U}_m,$$

which is obtained by Algorithms 3.15 and 3.12. This is of great interest, because usually we would like to solve this system by a Krylov–subspace based method.
The following lemma will give an answer. We will show that the sequence of coupling systems arising from the nested sequence of splittings can be obtained from a block LU decomposition of S_c after a similarity transformation with the product of $U_0 \cdots U_{m-1}$

Lemma 3.18 *Let $A = S - FG$ with $A, S \in \mathrm{GL}(n, \mathbb{F})$ and $F, G^T \in \mathrm{M}(n \times r, \mathbb{F})$. Using the notation of (3.4)-(3.8) we assume that for $k = 0, 1, \ldots, m-1$ the matrix $\tilde{U}_k$ is chosen such that $\tilde{S}_{c,k}$ is nonsingular. Then $S_0, \ldots, S_m$ and $S_{c,0}, \ldots, S_{c,m}$ are also nonsingular. In this case define for any $U_k = \left[\tilde{U}_k, \hat{U}_k\right]$ of size $r_k \times r_k$ the augmented matrix U^k of size $r \times r$ by*

$$U^k := \left(\begin{array}{c|c} I_{r-r_k} & \mathrm{O} \\ \hline \mathrm{O} & U_k \end{array}\right)$$

and set $Q := U^0 \cdots U^{m-1}$. The product Q induces a column partitioning with columns of size $s_0, \ldots, s_{m-1}, r_m$. Then $Q^* S_c Q$ has a decomposition

$$Q^* S_c Q = LDR,$$

where

$$L = \begin{pmatrix} \tilde{S}_{c,0} & & & & \mathrm{O} \\ \tilde{B}_{1,0} & \ddots & & & \\ \vdots & \ddots & \ddots & & \\ \tilde{B}_{m-1,0} & \cdots & \tilde{B}_{m-1,m-2} & \tilde{S}_{c,m-1} & \\ B_{m,0} & \cdots & B_{m,m-2} & B_{m,m-1} & I \end{pmatrix}, \tag{3.19}$$

$$D = \mathrm{diag}\left(\tilde{S}_{c,0}^{-1}, \ldots, \tilde{S}_{c,m-1}^{-1}, S_{c,m}\right), \tag{3.20}$$

$$R = \begin{pmatrix} \tilde{S}_{c,0} & \tilde{B}_{0,1} & \cdots & \tilde{B}_{0,m-1} & B_{0,m} \\ & \ddots & \ddots & \vdots & \vdots \\ & & \ddots & \tilde{B}_{m-2,m-1} & B_{m-2,m} \\ & & & \tilde{S}_{c,m-1} & B_{m-1,m} \\ \mathrm{O} & & & & I \end{pmatrix}. \tag{3.21}$$

For any $0 \leqslant l < k < m$, $\tilde{B}_{kl}$ and $\tilde{B}_{lk}$ satisfy

$$\tilde{B}_{kl} = \tilde{U}_k^T \hat{U}_{k-1}^T \cdots \hat{U}_l^T S_{c,l} \tilde{U}_l, \; \tilde{B}_{lk} = \tilde{U}_l^T S_{c,l} \hat{U}_l^T \cdots \hat{U}_{k-1} \tilde{U}_k. \tag{3.22}$$

For any $l = 0, \ldots, m-1$, B_{ml} and B_{lm} satisfy

$$B_{ml} = \hat{U}_{m-1}^T \cdots \hat{U}_l^T S_{c,l} \tilde{U}_l, \; B_{lm} = \tilde{U}_l^T S_{c,l} \hat{U}_l^T \cdots \hat{U}_{m-1}. \tag{3.23}$$

Proof:
The proof consist of three steps.
As first step we will show the nonsingularity of $S_0, \ldots, S_m$, $S_{c,0}, \ldots, S_{c,m}$.
As second step we will show that

$$U_k^T S_{c,k} U_k = \begin{pmatrix} \tilde{S}_{c,k} & \mathrm{O} \\ \hat{U}_k^T S_{c,k} \tilde{U}_k & I \end{pmatrix} \begin{pmatrix} \tilde{S}_{c,k}^{-1} & \mathrm{O} \\ \mathrm{O} & S_{c,k+1} \end{pmatrix} \begin{pmatrix} \tilde{S}_{c,k} & \tilde{U}_k^T S_{c,k} \hat{U}_k \\ \mathrm{O} & I \end{pmatrix}. \tag{3.24}$$

And finally as third step we will prove the LDR decomposition of $Q^* S_c Q$ with L, D, R from (3.19)–(3.23) by induction on m using (3.24).

Step 1: We will first show the nonsingularity of $S_0,\dots,S_m$, $S_{c,0},\dots,S_{c,m}$. For this we note that if $A = S - FG$ with a nonsingular matrix S, then A is nonsingular, if and only if the coupling system $S_c = I - GS^{-1}F$ is nonsingular. We have already pointed this fact out when introducing the Sherman–Morrison–Woodbury formula in (1.4).
For $m = 0$ there is nothing to show, since $S_0 = S, A$ are assumed to be nonsingular, so $S_{c,0} = S_c$ is nonsingular.
If $S_0,\dots,S_{m-1}$, $S_{c,0},\dots,S_{c,m-1}$ are nonsingular for some $m > 0$ and $\tilde{U}_{m-1}$ is chosen such that $\tilde{S}_{c,m-1}$ is nonsingular, then S_m must be nonsingular, too. This follows from the fact that $\tilde{S}_{c,m-1}$ is the coupling system with respect to the splitting $S_m = S_{m-1} - \tilde{F}_m\tilde{G}_m$ and from the nonsingularity of $S_{m-1}, \tilde{S}_{c,m-1}$.
But if S_m is nonsingular, then $S_{c,m}$ will also be nonsingular. This follows from the fact that $S_{c,m}$ is the coupling system with respect to the splitting $A = S_m - F_mG_m$ and from the nonsingularity of A, S_m.

Step 2: For (3.24) we have to show that

$$S_{c,k+1} = \hat{U}_k^T S_{c,k}\hat{U}_k - \hat{U}_k^T S_{c,k}\tilde{U}_k\tilde{S}_{c,k}^{-1}\tilde{U}_k^T S_{c,k}\hat{U}_k.$$

But by (3.6) and (3.8) we have that

$$\begin{aligned} S_{c,k+1} &= I - G_{k+1}S_{k+1}^{-1}F_{k+1} \\ &= I - \hat{U}_k^T G_k(S_k^{-1} + S_k^{-1}\tilde{F}_k\tilde{S}_{c,k}^{-1}\tilde{G}_kS_k^{-1})F_k\hat{U}_k \\ &= \hat{U}_k^T(I - G_kS_k^{-1}F_k)\hat{U}_k - \hat{U}_k^T G_kS_k^{-1}F_k\tilde{U}_k\tilde{S}_{c,k}^{-1}\tilde{U}_kG_kS_k^{-1}F_k\hat{U}_k \\ &= \hat{U}_k^T S_{c,k}\hat{U}_k - \hat{U}_k^T S_{c,k}\tilde{U}_k\tilde{S}_{c,k}^{-1}\tilde{U}_kS_{c,k}\hat{U}_k. \end{aligned}$$

Step 3: The proof of the LDR decomposition of Q^*S_cQ will be done by induction on m. For $m = 0$ this is just (3.24) with $m = k = 0$. Now assume that (3.19)–(3.23) hold for some m. To distinguish between step m and $m + 1$ we assume that $L \equiv L_m$, $D \equiv D_m$, $R \equiv R_m$, $Q \equiv Q_m$. We will show that (3.19)–(3.23) are true for m replaced by $m + 1$.

By (3.24) we have

$$(3.25)\qquad U_m^T S_{c,m}U_m = \begin{pmatrix} \tilde{S}_{c,m} & O \\ B_{m+1,m} & I \end{pmatrix}\begin{pmatrix} \tilde{S}_{c,m}^{-1} & O \\ O & S_{c,m+1} \end{pmatrix}\begin{pmatrix} \tilde{S}_{c,m} & B_{m,m+1} \\ O & I \end{pmatrix}.$$

From this it follows that

$$\begin{aligned} Q_{m+1}^T S_cQ_{m+1} &= (U^m)^T(Q_m^TS_cQ_m)U^m \\ &= ((U^m)^TL_mU^m)((U^m)^TD_mU^m)((U^m)^TR_mU^m). \end{aligned}$$

$(U^m)^TL_mU^m$ will be the matrix from (3.19) with $[B_{m,0},\dots,B_{m,m-2},B_{m,m-1}]$ replaced by $\begin{bmatrix} \tilde{B}_{m,0} & \cdots & \tilde{B}_{m,m-2} & \tilde{B}_{m,m-1} \\ B_{m+1,0} & \cdots & B_{m+1,m-2} & B_{m+1,m-1} \end{bmatrix}$. Analogous arguments can be applied to $(U^m)^TR_mU^m$.
To complete the proof we note that $(U^m)^TD_mU^m = \operatorname{diag}\left(\tilde{S}_{c,0}^{-1},\dots,\tilde{S}_{c,m-1}^{-1},U_m^TS_{c,m}U_m\right)$ and $U_m^TS_{c,m}U_m$ can be replaced by the decomposition (3.25). □

Remark: It follows from Lemma 3.18 that we get the coupling systems $\tilde{S}_{c,0}, \dots, \tilde{S}_{c,m-1}, S_{c,m}$ by performing a block LU–decomposition of $U^* S_c U$.

From the practical point of view we can easily obtain the sequence of resulting coupling systems $S_{c,k}$ throughout the process. Given a coupling system $S_{c,k}$ at some stage of the nested generation we apply a similarity transformation with U_k to $S_{c,k}$:

$$S_{c,k} \to U_k^* S_{c,k} U_k.$$

After this similarity transformation we partition the new matrix with respect to $U_k = \left[\tilde{U}_k, \hat{U}_k\right]$:

$$\begin{bmatrix} \tilde{U}_k^* \\ \hat{U}_k^* \end{bmatrix} S_{c,k} \begin{bmatrix} \tilde{U}_k & \hat{U}_k \end{bmatrix} = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix}.$$

By Lemma 3.18 it follows that $T_{11} = \tilde{S}_{c,k}$ is the small coupling system while the Schur–complement $T_{22} - T_{21} T_{11}^{-1} T_{12} = S_{c,k+1}$ will be the new coupling system. For $S_{c,k+1}$ one can apply the same strategy again.
We will illustrate this by an example.

Example 3.26 *In Example 3.1 the initial coupling system will be*

$$S_{c,0} = \left(\begin{array}{c|ccc} 1 & -9.9951 \cdot 10^{-1} & -5.0024 \cdot 10^{-1} & 0 \\ \hline -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -9.9951 \cdot 10^{-1} \\ 0 & -4.8852 \cdot 10^{-4} & -4.9976 \cdot 10^{-1} & 1 \end{array}\right).$$

Using the first choice $\tilde{F}, \tilde{G}$ we need not perform a similarity transformation to S_c since $\tilde{F}, \tilde{G}$ is the first column/row for F, G. We can choose $U = I$ and $U^* S_c U = S_c$. We obtain as small coupling system $\tilde{S}_{c,0} = 1$ and as remaining coupling system

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -9.9951 \cdot 10^{-1} \\ -4.8852 \cdot 10^{-4} & -4.9976 \cdot 10^{-1} & 1 \end{pmatrix} - \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} -9.9951 \cdot 10^{-1} & -5.0024 \cdot 10^{-1} & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 4.8852 \cdot 10^{-4} & -5.0024 \cdot 10^{-4} & 0 \\ 0 & 1 & -9.9951 \cdot 10^{-1} \\ -4.8852 \cdot 10^{-4} & -4.9976 \cdot 10^{-1} & 1 \end{pmatrix} = S_{c,1}.$$

By Lemma 3.18 it is not surprising that the remaining coupling system which has been derived here coincides with that of Example 3.3.

In this section we have shown, that if $\tilde{U}_k$ is chosen such that $\tilde{S}_{c,k}$ is nonsingular in any step, then the nested sequence of splittings exists. The corresponding remaining coupling systems $S_{c,k+1}$ can be obtained from their predecessor by a similarity transformation with U_k and then taking the Schur–complement.
The question, which has not been discussed so far is the choice of $\tilde{U}_k$. Of course we have some hints what conditions should be fulfilled by $\tilde{U}_k$. First it should ensure that $\tilde{S}_{c,k}$

is nonsingular and second it should improve the properties of the new coupling system $S_{c,k+1}$. Example 3.3 has shown that different choices of $\tilde{U}_k$ may lead to completely different conditioned coupling system. In one case, the condition number only has slightly changed, while in the other case it drastically decreased. This topic will be discussed in the next section.

3.5 Modifying Schur–Complements by A Priori Orthogonal Transformations

In this section we will discuss the following problem:
Let $M \in \mathrm{GL}\,(r, \mathbb{F})$. Which kind of orthogonal matrix $U = [\tilde{U},\ \hat{U}]$ with $\tilde{U} \in \mathrm{M}\,(r \times s, \mathbb{F})$, $\hat{U} \in \mathrm{M}\,(r \times (r-s), \mathbb{F})$ is suitable to improve the properties of the Schur–complement of the transformed matrix U^*MU. I.e., we are interested in improving the properties of

$$B = \hat{U}^*M\hat{U} - \hat{U}M\tilde{U}(\tilde{U}^*M\tilde{U})^{-1}\tilde{U}^*M\hat{U}. \tag{3.27}$$

This question will be of great interest for the construction of nested splittings.

Independent on the choice of U we have for any $\tilde{U}$ such that $\tilde{U}^*M\tilde{U}$ is nonsingular

$$\|B^{-1}\|_2 \leqslant \|M^{-1}\|_2. \tag{3.28}$$

This follows immediately from the fact that B^{-1} is the lower right diagonal block of $(U^*MU)^{-1}$.

A very simple choice for $\tilde{U}_k$ can be the following one. Denote by $\lambda_1, \ldots, \lambda_r$ the eigenvalues of M. Assume that $\tilde{U}$ corresponds to an invariant subspace of M with respect to the eigenvalues $\lambda_1, \ldots, \lambda_s$. For any orthogonal matrix $U = [\tilde{U},\ \hat{U}]$ the Schur–complement B from (3.27) will have the eigenvalues $\lambda_{s+1}, \ldots, \lambda_r$. It is clear that by this choice of $\tilde{U}_k$ the condition number of B will be less than or equal to the condition number of M.

More interesting will be the following result which uses a linear combination of two orthogonal eigenvectors.

Lemma 3.29 *Let $M \in \mathrm{GL}\,(r, \mathbb{F})$ and suppose that there exist $u, v \in \mathbb{F}^r$ such that $Mu = \lambda u, Mv = \mu v$ and $[u,v]^T[u,v] = I$. We define $\tilde{U}$ by $\tilde{U} = cu + \sqrt{1-c^2}v$, where $c \in [0,1]$ is chosen such that*

$$c^2\lambda + (1-c^2)\mu \neq 0.$$

Then $\tilde{U}^*M\tilde{U} = c^2\lambda + (1-c^2)\mu$ and for any orthogonal (unitary) $U = \left[\tilde{U}, \hat{U}\right]$ the set of eigenvalues of B will be the set of eigenvalues of M with λ, μ replaced by $\nu = \frac{\lambda\mu}{c^2\lambda+(1-c^2)\mu}$.

Proof:
We consider an orthogonal (unitary) matrix W, such that the first two columns of W are

u and v. An equivalence transformation W^*MW will produce a matrix of following form.

$$W^*MW = \begin{pmatrix} \lambda & 0 & M_{13} \\ 0 & \mu & M_{23} \\ 0 & 0 & M_{33} \end{pmatrix}.$$

We set $s = \sqrt{1-c^2}$ and $G_2 = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$. We apply a further equivalence transformation to W^*MW using the block diagonal orthogonal matrix G which has G_2 as upper left 2×2 block and I in the other block diagonal position. This will lead to the following matrix.

$$G^*W^*MWG = \begin{pmatrix} c^2\lambda + s^2\mu & cs(\lambda-\mu) & cM_{13} - sM_{23} \\ cs(\lambda-\mu) & s^2\lambda + c^2\mu & sM_{13} + cM_{23} \\ 0 & 0 & M_{33} \end{pmatrix}.$$

Since $c^2\lambda + s^2\mu$ is assumed to be different from 0, we can take the Schur–complement with respect to the upper left 1×1 block:

$$\begin{pmatrix} \frac{\lambda\mu}{c^2\lambda+s^2\mu} & \hat{M}_{23} \\ 0 & M_{33} \end{pmatrix}.$$

If U is any matrix such that the first column of U is that of WG, then the columns 2 to r of U can be obtained from columns 2 to r of WG by multiplication with an orthogonal (unitary) matrix Q of size $(r-1) \times (r-1)$ from the right. But this will be an orthogonal similarity transformation to B which does not change the eigenvalues of B. □

Remark: Note that the construction of $\tilde{U}$ is possible for any normal matrix. Otherwise one cannot guarantee to find a pair of orthogonal eigenvectors. One has to be careful how to choose c, since the function

$$f_{\mu,\lambda}(t) = \frac{\lambda\mu}{t\lambda + (1-t)\mu}$$

has poles, if the convex hull of λ and μ contains 0.

It is clear that one can generalize Lemma 3.29 if one can find $2k$ piecewise orthogonal eigenvectors and combines k times pairs of eigenvectors. For the general case it is not clear if choosing eigenvectors of the matrix is advisable. At least some success has been made for the GMRES method in [62] when augmenting the Krylov subspace with eigenvectors corresponding to eigenvalues with small modulus. Another interesting approach using low rank modifications has been made in [53]. This might be an interesting approach for future works.

For the symmetric positive definite case we can derive optimal orthogonal transformations, which will be discussed next.

3.5.1 Optimal Conditioning Schur–Complements of Symmetric Positive Definite Matrices by A Priori Orthogonal Transformations

In the case when $M \in \mathrm{GL}\,(r,\mathbb{F})$ is a symmetric (Hermitian) positive definite matrix $M \in \mathrm{GL}\,(r,\mathbb{F})$ we can determine the optimal choice of an orthogonal (unitary) $U = [\tilde{U},\ \hat{U}]$, $\tilde{U} \in \mathrm{M}\,(r \times s, \mathbb{F})$, with respect to the condition number, i.e. find $U = [\tilde{U},\ \hat{U}]$ such that

$$\mathrm{cond}_2(B) = \mathrm{cond}_2(\hat{U}^* M \hat{U} - \hat{U} M \tilde{U} (\tilde{U}^* M \tilde{U})^{-1} \tilde{U}^* M \hat{U}) \overset{!}{=} \min\,. \tag{3.30}$$

In other words: We look for an orthogonal (unitary) matrix U which minimizes the condition number of the Schur–complement of U^*MU with respect to the upper left $s \times s$ block. The following lemma shows, that skillfully combining the eigenvectors of the smallest eigenvalues and largest eigenvalues pairwise will lead to the optimal choice.

Theorem 3.31 *Let $M \in \mathrm{GL}\,(r,\mathbb{F})$, $M = M^*$ positive definite. Let s be a number satisfying $0 < s < r/2$. Denote by $\lambda_1 \geqslant \ldots \geqslant \lambda_r > 0$ the set of eigenvalues of M and by $v_1,\ldots,v_r$ a corresponding set of orthonormal eigenvectors. We define $u_i = c_i v_i + \sqrt{1-c_i^2}\, v_{r-i+1}$, where $c_i \in [0,1]$ is chosen such that*

$$\nu_i = \frac{\lambda_i \lambda_{r-i+1}}{c_i^2 \lambda_i + (1-c_i^2)\lambda_{r-i+1}} \in [\lambda_{r-s},\ \lambda_{s+1}]$$

for all $i = 1,\ldots,s$. Let $U = [\tilde{U},\ \hat{U}]$ be orthogonal (unitary), such that $\tilde{U} = [u_1,\ldots u_s]$. Then the eigenvalues of B be from (3.30) are

$$\nu_1,\ldots,\nu_s, \lambda_{s+1},\ldots,\lambda_{r-s} \tag{3.32}$$

and B satisfies in the sense of quadratic forms

$$\lambda_{r-s} I \leqslant B \leqslant \lambda_{s+1} I. \tag{3.33}$$

This choice is optimal in the following sense: for any orthogonal (unitary) $W = [\tilde{W}, \hat{W}]$ with same partitioning as U it follows from

$$\gamma I \leqslant \hat{W}^* M \hat{W} - \hat{W} M \tilde{W} (\tilde{W}^* M \tilde{W})^{-1} \tilde{W}^* M \hat{W} \leqslant \Gamma I$$

that $\gamma \leqslant \lambda_{r-s}$, $\lambda_{s+1} \leqslant \Gamma$.

Proof:
First of all we point out that by applying Lemma 3.29 s times with the pairs $(v_1,v_r),(v_2,v_{r-1}),\ldots,(v_s,v_{r-s+1})$ we get a Schur–complement B where the eigenvalues $\lambda_1,\ldots,\lambda_s$, $\lambda_{r-s+1},\ldots,\lambda_r$ of M have been replaced by $\nu_1,\ldots,\nu_s$ while the remaining eigenvalues are those of M. This shows (3.32). But the choice of $c_1,\ldots,c_s$ ensures that the s new eigenvalues lie between the remaining eigenvalues $\lambda_{s+1},\ldots,\lambda_{r-s}$. From this it follows that

$$\lambda_{r-s} I \leqslant B \leqslant \lambda_{s+1}.$$

Finally we will show the optimality. For any orthogonal (unitary) $W = \left[\tilde{W}, \hat{W}\right]$ with the same partitioning as U we have that $D^{-1} = \left[\hat{W}^* M \hat{W} - \hat{W} M \tilde{W} (\tilde{W}^* M \tilde{W})^{-1} \tilde{W}^* M \hat{W}\right]^{-1}$ is the $(r-s) \times (r-s)$ diagonal block of $(W^* M W)^{-1}$ in the lower right corner. Using the Rayleigh quotient characterization of eigenvalues [41], p. 411, we have that

$$\frac{1}{\lambda_{r-s}} = \min_{\dim(S)=r-s} \max_{x \in S \setminus \{0\}} \frac{x^* M^{-1} x}{x^* x}, \quad \frac{1}{\lambda_{s+1}} = \max_{\dim(S)=r-s} \min_{x \in S \setminus \{0\}} \frac{x^* M^{-1} x}{x^* x}.$$

The space which is spanned by the columns of $\hat{W}$ is one possible choice for S, so $\frac{1}{\gamma} \geqslant \frac{1}{\lambda_{r-s}}$, $\frac{1}{\Gamma} \leqslant \frac{1}{\lambda_{s+1}}$, from which the assertion of the theorem follows. $\square$

Summary

In this chapter we have shown that a nested use of the Sherman–Morrison–Woodbury formula can be employed to improve the properties of the remaining coupling system $S_{c,m}$. Note that for substructuring methods the improvement of the Schur–complement is usually done by construction of preconditioners. Here, for Divide & Conquer methods the successive modification of the initial splitting gives an additional way to achieve an improved coupling system. The successive modification can be read as adaptively constructing a sequence of preconditioners $S = S_0, S_1, \ldots, S_m$ where the rank of the remaining matrix is reduced at the same time. This strategy can be summarized in the following table.

Nested Divide & Conquer			
Level	Current Splitting	remaining size	Current Coupling System
0	initial splitting	r_0	initial coupling system
	$A = S_0 - F_0 G_0$		$S_{c,0}$
1	current splitting +rank s_0	$r_0 - s_0$	Schur-complement of the current coupling system after transformation
	$A = S_0 - F_0 G_0$ $\downarrow$ $A = S_1 - F_1 G_1$		$S_{c,0}$ $\downarrow$ $S_{c,1}$
$\vdots$			
k	current splitting +rank s_{k-1}	$r_0 - s_0 - \cdots s_{k-1}$	Schur-complement of the current coupling system after transformation
	$A = S_{k-1} - F_{k-1} G_{k-1}$ $\downarrow$ $A = S_k - F_k G_k$		$S_{c,k-1}$ $\downarrow$ $S_{c,k}$

We can interpret this strategy as a compromise between a direct solution and an iterative solution of the coupling system. This problem was part of question 3 from our list of questions on page 10. It has turned out that the related coupling systems $\tilde{S}_{c,0}, \ldots, \tilde{S}_{c,m-1}, S_{c,m}$

can be viewed as diagonal blocks in a block LU–decomposition of the original coupling system, after a suitable pre– and post multiplication. In general it is still open what transformation should be used for the pre– and post multiplication.

The interpretation as LU decomposition leads to the question in which way Divide & Conquer methods are related to block ILU–decompositions and algebraic multigrid methods. This will be discussed in the next chapter.

Chapter 4

Relations to Algebraic Multigrid Methods

In this chapter we will demonstrate the close relationship between the nested use of the Sherman–Morrison–Woodbury formula (1.4), (1.9), especially their nested use in Chapter 3, and algebraic multigrid methods.
Recall that we have two equivalent low rank modification formulas for writing the inverse of $A = S - FG$, where $A, S \in \mathrm{GL}(n, \mathbb{F})$, $G, F^T \in \mathrm{M}(n \times r, \mathbb{F})$. The first one, referred as Sherman–Morrison–Woodbury formula (1.3) can be seen as Schur–complement approach for a suitably extended system. The second one in (1.9), which assumes the existence of a left inverse matrix $H \in \mathrm{M}(r \times n, \mathbb{F})$ such that $HF = I$ can be viewed as a two level iteration for A.

These two different points of view and their application will be discussed in this chapter.

We will start with some general results from algebraic multigrid methods.

Definition 4.1 *Consider a linear system $Ax = b$, $A \in \mathrm{GL}(n, \mathbb{F})$, $x, b \in \mathbb{F}^n$ and a linear operator $B \in \mathrm{M}(n \times n, \mathbb{F})$.*
Then a linear **iteration–scheme** *is given by the iterates $\left(x^{(k)}\right)_{k \in \mathbb{N}}$, where*
$x^{(0)} \in \mathbb{F}^n$ is an initial guess, $\quad x^{(k+1)} = x^{(k)} - B(Ax^{(k)} - b), \quad k = 0, 1, 2, 3, \ldots$
The matrix $I - BA$ is called **linear iteration operator**.

Some well–known results are given in the following Lemma (see [41], [18]):

Lemma 4.2 *Consider a linear system $Ax = b$, $A \in \mathrm{GL}(n, \mathbb{F})$, $x, b \in \mathbb{F}^n$ and a linear operator $B \in \mathrm{M}(n \times n, \mathbb{F})$. Then*

- $x^{(k)} - x = (I - BA)^k (x^{(0)} - x)$
- *The sequence $\left(x^{(k)}\right)_{k \in \mathbb{N}}$ converges to x for any $x^{(0)} \in \mathbb{F}^n$ if and only if $\rho(I - BA) < 1$. $\rho(I - BA)$ denotes the largest eigenvalue in modulus of $I - BA$ (see Definition 2.17).*

We will now explain an algebraic multigrid iteration.

Definition 4.3 *Consider a non–singular linear operator $A : \mathbb{F}^n \longrightarrow \mathbb{F}^n$ and a sequence of $l \geqslant 2$ nested spaces $\mathbb{F}^n \equiv \mathbb{F}^{n_1} \supset \mathbb{F}^{n_2} \supset \ldots \supset \mathbb{F}^{n_l} \not\equiv \{0\}$.*
For each $k \in \{1, \ldots, l-1\}$ we consider linear operators $A_k, \Sigma_k : \mathbb{F}^{n_k} \longrightarrow \mathbb{F}^{n_k}$ defined as follows. $A_l : \mathbb{F}^{n_l} \longrightarrow \mathbb{F}^{n_l}$, where $A_1 = A$ and each A_k is non–singular, $k = 1, \ldots, l$.
We consider further linear restriction operators $R_{k+1,k} : \mathbb{F}^{n_k} \longrightarrow \mathbb{F}^{n_{k+1}}$ and prolongation operators $P_{k,k+1} : \mathbb{F}^{n_{k+1}} \longrightarrow \mathbb{F}^{n_k}$ for $k = 1, \ldots, l-1$.
Then an **algebraic multigrid operator** *$M \equiv M_1$ is recursively defined by:*

$$(4.4) \qquad \begin{array}{rcl} I - M_k A_k &=& (I - P_{k,k+1} M_{k+1} R_{k+1,k} A_k)^{m_k} (I - \Sigma_k A_k)^{\nu_k}, \ k < l \\ M_l &=& A_l^{-1}, \end{array}$$

where ν_k, m_k are natural numbers, $m_k > 0$, $\nu_k > 0$.
The corresponding iteration

$$x^{(t+1)} := x^{(t)} - M(Ax^{(t)} - b), \quad x^{(0)} \text{ initial guess}$$

is called **algebraic multigrid iteration***(AMG–iteration).*
Σ_k is called **(pre–) smoother***.*
A_l is called **coarse grid correction***.*

Remark:

- In [47],[48],[49], all m_k are identical and chosen as 1 or 2.
  For $m_k \equiv 1$ the iteration scheme is often called V-cycle.
  For $m_k \equiv 2$ the iteration scheme is often called W-cycle.
- Multigrid methods are often used in the numerical treatment of partial differential equations. There the subspaces $\mathbb{F}^{n_k}$, $k = 1, \ldots, l$ correspond to hierarchical grids. For the matrix A_k often a model–dependent linear operator is used.
  Restrictions and prolongations are usually chosen with respect to the special grid hierarchy.
- Each A_k can be viewed as approximation to A on a suitable subspace, i.e.

  $$R_{k+1,k} A_k P_{k,k+1} \approx A_{k+1}$$

  In fact, if $R_{k+1,k} A_k P_{k,k+1} = A_{k+1}$ for any $0 < k < l-1$, then the nested iteration from Definition 4.3 can already be read as a product iteration for A_k itself, which will be shown in the next lemma. This generalizes a result of [85] for the symmetric positive definite case and $R^*_{k+1,k} = P_{k,k+1}$, $R^*_{k,k+1} R_{k+1,k} = I$.

Lemma 4.5 *Let $k \leqslant n, A \in \mathrm{GL}(n, \mathbb{F}), B \in \mathrm{GL}(k, \mathbb{F})$, $P, R^T \in \mathrm{M}(n \times k, \mathbb{F})$, $M, N_1, \ldots, N_l \in \mathrm{M}(k \times k, \mathbb{F})$. Let $I - MB = (I - N_1 B) \cdots (I - N_l B)$. If $B = RAP$, then*

$$I - PMRA = (I - PN_1 RA) \cdots (I - PN_l RA)$$

Using the notation of Definition 4.3 we have for $1 < k < l$*: If* $R_{k,k-1}A_{k-1}P_{k-1,k} = A_k$*, then*

$$(4.6) \quad \begin{aligned} I - P_{k-1,k}M_kR_{k,k-1}A_{k-1} &= (I - P_{k-1,k}P_{k,k+1}M_{k+1}R_{k+1,k}R_{k,k-1}A_{k-1})^{m_k} \\ &\quad \cdot(I - P_{k-1,k}\Sigma_{1,k}R_{k,k-1}A_{k-1})^{\nu_k} \end{aligned}$$

Proof:
We prove the first part by induction on l.
For $l = 1$ this is obviously true.
Assume that the assertion holds for some $l \geqslant 1$. Then

$$\begin{aligned} I - PMRA &= I - P\,[I - (I - N_1B)\cdots(I - N_lB)(I - N_{l+1}B)]\,B^{-1}RA \\ &= I - P\,[I - (I - N_1B)\cdots(I - N_lB)]\,B^{-1}RA \\ &\quad -P(I - N_1B)\cdots(I - N_lB)N_{l+1}RA \\ &\overset{Ind.}{=} (I - PN_1RA)\dots(I - PN_lRA) \\ &\quad -(I - PN_1RA)\cdots(I - PN_lRA)PN_{l+1}RA \\ &= (I - PN_1RA)\cdots(I - PN_lRA)(I - PN_{l+1}RA) \end{aligned}$$

The second part then follows immediately. $\square$

Successively applied, (4.6) will finally end in a product iteration of the form

$$I - M_1A = (I - P_1N_1R_1A)\cdots(I - P_rN_rR_rA),$$

where $r = \nu_1 + m_1(\nu_2 + m_2(\cdots\nu_{l-2} + m_{l-2}(\nu_{l-1} + m_{l-1})\cdots))$ is the total number of iteration steps. Each P_i, R_i can be written as product of the form $P_i = P_{12}P_{23}\cdots P_{k_i,k_i+1}$, $R_i = R_{k_i+1,k_i}R_{k_i,k_i-1}\cdots R_{21}$. N_i is either Σ_{k_i+1} or M_l, if $k_i = l-1$.

This shows that the nested iteration from Definition 4.3 can be lifted to a product iteration for A if the induced system matrix on the smaller space is chosen to be the Ritz approximation $B = RAP$. In other words, for algebraic multilevel methods with Ritz approximation as coarse grid system any nested iteration can be read as multiplicative iteration for the initial system using the corresponding products of restriction and prolongation operators. For the symmetric positive case this was already shown in [85].

4.1 Incomplete block LU Decompositions as Algebraic Multigrid Method

We will briefly explain how a block LU decomposition and also an incomplete block LU decomposition can be read as algebraic multigrid method. Any nonsingular matrix $A \equiv \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \in \mathrm{GL}\,(n, \mathbb{F})$ with nonsingular A_{11} can be written as

$$(4.7)\ A = \begin{pmatrix} I & O \\ A_{21}A_{11}^{-1} & I \end{pmatrix}\begin{pmatrix} A_{11} & O \\ O & S_{22} \end{pmatrix}\begin{pmatrix} I & A_{11}^{-1}A_{12} \\ O & I \end{pmatrix}, \text{ where } S_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}.$$

It is well–known (see e.g. [20]) that a block LU–decomposition can read as exact algebraic 2–level method. This can be seen from the fact that

$$
(4.8)\qquad O = (I - \underbrace{\begin{pmatrix} -A_{11}^{-1}A_{12} \\ I \end{pmatrix}}_{P} S_{22}^{-1} \underbrace{\begin{pmatrix} -A_{21}A_{11}^{-1} & I \end{pmatrix}}_{R} A)(I - \underbrace{\begin{pmatrix} A_{11}^{-1} & O \\ O & O \end{pmatrix}}_{\Sigma} A).
$$

For a block ILU decomposition one typically replaces $A_{11} \in \mathrm{GL}\,(r, \mathbb{F})$ by some nonsingular approximations L_{11}, U_{11}, D_{11} and S_{22} by D_{22}. This leads to the following matrix

$$
(4.9)\qquad \underbrace{\begin{pmatrix} I & O \\ A_{21}L_{11}^{-1} & I \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} D_{11} & O \\ O & D_{22} \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} I & U_{11}^{-1}A_{12} \\ O & I \end{pmatrix}}_{U}
$$

and defines in a natural way an algebraic 2–level method by setting

$$
R = \begin{pmatrix} -A_{21}L_{11}^{-1} & I \end{pmatrix},\ P = \begin{pmatrix} -U_{11}^{-1}A_{12} \\ I \end{pmatrix},\ \Sigma = \begin{pmatrix} A_{11}^{-1} & O \\ O & O \end{pmatrix}.
$$

A special case is the choice $D_{22} = RAP = S_{22} + E_{22}$, where $E_{22} = A_{21}(L_{11}^{-1} - A_{11}^{-1})A_{11}(U_{11}^{-1} - A_{11}^{-1})A_{12}$. In this case Lemma 4.5 is applicable, i.e. when successively applying an incomplete block LU decomposition with $D_{22} = S_{22} + E_{22}$ the iteration can be lifted to a product iteration for the initial matrix A.

4.2 Theoretical Results for the Positive Definite Case

Let us consider the case when A is symmetric (Hermitian) and positive definite. We will cite two results for algebraic multigrid methods. The first one [45], [46] discusses spectral equivalence properties for incomplete LU–decompositions. The second one [72] discusses the convergence rate of the 2–level scheme (4.3). These two results can be applied to the two low rank modification formulas (1.4),(1.9).

The first result [45], [46] will be a result concerning the approximation properties of incomplete LU–decompositions.
For a pair A, B of two symmetric (Hermitian) positive definite matrices a typical way to get estimates for $\mathrm{cond}_2(B^{-1/2}AB^{-1/2})$ is comparing A and B in the sense of quadratic forms, i.e., we have to find $\Gamma, \gamma > 0$ such that

$$
\gamma B \leqslant A \leqslant \Gamma B.
$$

From this it follows that $\mathrm{cond}_2(B^{-1/2}AB^{-1/2}) \leqslant \frac{\Gamma}{\gamma}$.
Here we want to approximate A by a block ILU–decomposition. The question is, how $\gamma, \Gamma > 0$ have to be chosen such that

$$
\gamma LDL^* \leqslant A \leqslant \Gamma LDL^*
$$

The condition number plays a key role for Krylov–subspace methods in the positive definite case. We cite a result from [45], [46] on so–called substructuring methods. Later this result will be applied to algebraic domain decomposition methods based on the nested use of the Sherman–Morrison–Woodbury formula.

We can apply (4.7) to a given symmetric (Hermitian) positive definite matrix $A \in \mathrm{GL}(n, \mathbb{F})$.

Analogously to the general case we can replace $A_{11} \in \mathrm{GL}(r, \mathbb{F})$ by some nonsingular approximations $L_{11}, U_{11} = L_{11}^*, D_{11}$ and S_{22} by D_{22}. This leads to the following matrix

$$(4.10)\qquad \underbrace{\begin{pmatrix} I & O \\ A_{21}L_{11}^{-1} & I \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} D_{11} & O \\ O & D_{22} \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} I & L_{11}^{-*}A_{12} \\ O & I \end{pmatrix}}_{L^*},$$

where at least D_{11}, D_{22} are assumed to be symmetric (Hermitian) positive definite. For the approximation properties of (4.10) the following result can be shown.

Theorem 4.11 *Let $A \in \mathrm{GL}(n, \mathbb{F})$ be symmetric (Hermitian), positive definite. Consider the block ILU–decomposition of A from (4.10). Consider $\gamma, \Gamma > 0$ such that*

$$\gamma D_{11} \leqslant A_{11} \leqslant \Gamma D_{11},\ \gamma D_{22} \leqslant S_{22} + E_{22} \leqslant \Gamma D_{22},$$

where E_{22} is defined by $E_{22} := A_{21}(A_{11}^{-1} - L_{11}^{-1})A_{11}(A_{11}^{-1} - L_{11}^{-*})A_{12}$ and S_{22} is taken from (4.7). Set $\mu = \rho(S_{22}^{-1}E_{22})$. Then the block ILU–factorization (4.10) satisfies

$$(4.12)\qquad \gamma\left(1 - \sqrt{\frac{\mu}{1+\mu}}\right) LDL^* \leqslant A \leqslant \Gamma\left(1 + \sqrt{\frac{\mu}{1+\mu}}\right) LDL^*.$$

Furthermore the condition number of $(LDL^*)^{-1}A$ can be estimated by

$$(4.13)\qquad \frac{\gamma}{\Gamma}\left(\sqrt{\mu} + \sqrt{1+\mu}\right)^2 \leqslant \frac{\lambda_{\max}((LDL^*)^{-1}A)}{\lambda_{\min}((LDL^*)^{-1}A)} \leqslant \frac{\Gamma}{\gamma}\left(\sqrt{\mu} + \sqrt{1+\mu}\right)^2,$$

where

$$(4.14)\qquad \sqrt{\frac{\mu}{1+\mu}} = \cos\angle(\mathrm{span}\begin{pmatrix} I \\ O \end{pmatrix},\ \mathrm{span}\begin{pmatrix} -L_{11}^{-*}A_{12} \\ I \end{pmatrix})$$

and the angle is taken with respect to the inner product defined by A.

Proof:
See [45]. □

Remark:
Note that $\mu = \rho(S_{22}^{-1}E_{22})$ implies that $\frac{\mu}{1+\mu} = \rho((S_{22} + E_{22}))^{-1}E_{22}) \equiv \sigma$. Thus we can replace $\sqrt{\frac{\mu}{1+\mu}}$ by $\sqrt{\sigma}$ in Theorem 4.11.

The second result from [72] can be used to get estimates for the convergence rate of the 2–level scheme in the symmetric positive definite case.

In the following we denote for $x \in \mathbb{F}^n, K \in \mathrm{GL}\,(n,\mathbb{F})$, $K = K^*$ positive definite, the K–norm by

$$\|x\|_K = \sqrt{x^* K x}.$$

For any $n \times n$ matrix A we denote by

$$\|A\|_K = \sup_{x \neq 0} \frac{\|Ax\|_K}{\|x\|_K}$$

the corresponding matrix norm.

Theorem 4.15 *Let $A \in \mathrm{GL}\,(n,\mathbb{F})$, $A = A^*$ positive definite, $H \in \mathrm{M}\,(n \times r, \mathbb{F})$,* rank $H =$ *r. H will be used as prolongation operator and its adjoint matrix H^* as restriction operator. Let $N \in \mathrm{M}\,(n \times n, \mathbb{F})$ be a smoother in the sense of Definition 4.3. Set $A_H = H^*AH, T = I - HA_H^{-1}H^*A$ and choose $\tilde{A}_H \in \mathrm{GL}\,(r,\mathbb{F})$ such that*

$$\|I - \tilde{A}_H^{-1} A_H\|_{A_H} \leqslant \eta < 1. \tag{4.16}$$

If there exists $\delta > 0$ such that

$$\|(I - NA)e\|_A^2 \leqslant \|e\|_A^2 - \delta \|T(I - NA)e\|_A^2 \tag{4.17}$$

for any $e \in \mathbb{F}^n$, then

$$\|(I - H\tilde{A}_H^{-1}H^*A)(I - NA)\|_A \leqslant \max\{\eta, \sqrt{\frac{1}{1+\delta}}\}. \tag{4.18}$$

Proof:
See [72]. □

4.3 Divide & Conquer as AMG

We will now apply the results from algebraic multigrid theory to the nested Divide & Conquer methods from Chapter 3. For this we will recall the properties of nested splittings obtained by nested Divide & Conquer algorithms. We consider a splitting

$$A = S - FG \equiv S_0 - F_0 G_0, \tag{4.19}$$

where $A, S \in \mathrm{GL}\,(n,\mathbb{F})$, $F, G^T \in \mathrm{M}\,(n \times r, \mathbb{F})$. Similar to Chapter 3 we assume that there exists a sequence of orthogonal (unitary) matrices $U_k = \left[\tilde{U}_k,\, \hat{U}_k\right] \in \mathrm{GL}\,(r_k,\mathbb{F}), k = 0\ldots, m-1$ such that $\tilde{U}_k \in \mathrm{M}\,(r_k \times s_k, \mathbb{F})$. Let $r_{k+1} = r_k - s_k, r_0 = r$.
In this case we can write A as a nested splitting in the following way: Define successively for $k = 0, \ldots, m-1$

$$\begin{array}{rcl} \tilde{F}_k &=& F\hat{U}_0 \cdots \hat{U}_{k-1}\tilde{U}_k, \\ \tilde{G}_k &=& \tilde{U}_k^* \hat{U}_{k-1}^* \cdots \hat{U}^*{}_0 G, \\ S_{k+1} &=& S_k - \tilde{F}_k \tilde{G}_k. \end{array} \tag{4.20}$$

Then we obtain

$$(4.21)\qquad \begin{aligned} A &= S_m - F_m G_m, \text{ where} \\ F_m &= F\hat{U}_0 \cdots \hat{U}_{m-1}, \\ G_m &= \hat{U}^*_{m-1} \cdots \hat{U}^*_0 G. \end{aligned}$$

By successive use of the Sherman–Morrison–Woodbury formula we obtain:

$$(4.22)\qquad \begin{aligned} S_0^{-1} &= S^{-1}, \\ S_{k+1}^{-1} &= S_k^{-1} + S_k^{-1}\tilde{F}_k(\underbrace{I - \tilde{G}_k S_k^{-1}\tilde{F}_k}_{\tilde{S}_{c,k}})^{-1}\tilde{G}_k S_k^{-1}, \ k = 0, \ldots, m-1, \\ A^{-1} &= S_m^{-1} - S_m^{-1} F_m(\underbrace{I - G_m S_m^{-1} F_m}_{S_{c,m}})^{-1} G_m S_m^{-1}. \end{aligned}$$

Now we will show that the Sherman–Morrison–Woodbury formula can be interpreted as Schur–complement for a suitable extended system, which shows the close relations to Schur–complement methods:

Theorem 4.23
Let $A, S \in \mathrm{GL}(n, \mathbb{F})$, $F, G^* \in \mathrm{M}(n \times r, \mathbb{F})$ *and assume that* $A = S - FG$. *Set* $S_c = I - GS^{-1}F$. *Then*

$$(4.24)\qquad Ax = b, \text{ if and only if } \begin{pmatrix} S & F \\ G & I \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Using the notation of (4.19)-(4.22) we have

$$(4.25)\ Ax = b, \text{ if and only if } \begin{pmatrix} S & \tilde{F}_0 & \cdots & \tilde{F}_{m-1} & F_m \\ \tilde{G}_0 & I & & & \\ \vdots & & \ddots & & \\ \tilde{G}_{m-1} & & & I & \\ G_m & & & & I \end{pmatrix}\begin{pmatrix} x \\ y_1 \\ \vdots \\ y_m \\ \hat{y}_m \end{pmatrix} = \begin{pmatrix} b \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

The augmented system has a block LU–decomposition of the form $\hat{L} \cdot \hat{D} \cdot \hat{R}$, *where*

$$(4.26)\hat{L} = \left(\begin{array}{c|c} S & \mathrm{O} \\ \hline \tilde{G}_0 & \\ \vdots & L \\ \tilde{G}_{m-1} & \\ G_m & \end{array}\right), \ \hat{D} = \operatorname{diag}\left(S^{-1}, D\right), \ \hat{R} = \left(\begin{array}{c|cccc} S & \tilde{F}_0 & \cdots & \tilde{F}_{m-1} & F_m \\ \hline \mathrm{O} & & & R & \end{array}\right)$$

and L, D, R *taken from (3.19)-(3.21).*

Proof:
(4.24) follows from

$$\begin{pmatrix} I & O \end{pmatrix} \begin{pmatrix} S & F \\ G & I \end{pmatrix}^{-1} \begin{pmatrix} I \\ O \end{pmatrix} = (S - FG)^{-1} = A^{-1}.$$

In the same way (4.25) follows from

$$\begin{aligned} & \begin{pmatrix} I & O & \cdots & O & O \end{pmatrix} \begin{pmatrix} S & \tilde{F}_0 & \cdots & \tilde{F}_{m-1} & F_m \\ \tilde{G}_0 & I & & & \\ \vdots & & \ddots & & \\ \tilde{G}_{m-1} & & & I & \\ G_m & & & & I \end{pmatrix}^{-1} \begin{pmatrix} I \\ O \\ \vdots \\ O \\ O \end{pmatrix} \\ = \; & (S - \tilde{F}_1\tilde{G}_1 - \ldots - \tilde{F}_{m-1}\tilde{G}_{m-1} - F_m G_m)^{-1} \\ = \; & A^{-1}. \end{aligned}$$

(4.26) immediately follows from Lemma 3.18. □

As a consequence of Theorem 4.23, we can get $x = A^{-1}b$ by the Schur–complement approach for the extended system:

$$(4.27) \quad x = (I,\ O) \begin{pmatrix} S & F \\ G & I \end{pmatrix}^{-1} \begin{pmatrix} b \\ 0 \end{pmatrix} = (I,\ -S^{-1}F) \begin{pmatrix} S^{-1} & O \\ O & S_c^{-1} \end{pmatrix} \begin{pmatrix} I \\ -GS^{-1} \end{pmatrix} b.$$

But this is just the Sherman–Morrison–Woodbury formula for inverting $A = S - FG$. Similarly we can proceed for a nested application of the Sherman–Morrison–Woodbury formula.

The main difference to substructuring methods is, that we extend the initial system with respect to the given splitting $A = S - W$ and the factorization $W = FG$, while substructuring methods only work with permutations.

Analogous to Schur–complement methods we can use the nested Divide & Conquer and a corresponding ILU–decomposition to define an approximation to the original system A.

4.3.1 The Positive Definite Case

For the special case of symmetric positive definite matrices we consider an approximate nested Divide & Conquer scheme similar to (3.4)–(3.8).

We assume that A, S are symmetric positive definite and $G = F^*$. It is clear that $G_m = F_m^*, \tilde{G}_k = \tilde{F}_k^*$.

Now we consider approximations $\Delta, \Sigma \in \mathrm{GL}\,(n, \mathbb{F})$ to S, where at least Δ should also be positive definite. This case may occur, if S is perturbed or if one uses an incomplete Cholesky decomposition for S.

First of all we define recursively a sequence Σ_k of approximations to $S_k,\ k=0,\dots,m$ and $\Delta_{c,k}$ to $\tilde{S}_{c,k},\ k=1,\dots,m$. Σ_k is not necessarily symmetric. Essentially we will replace S_k^{-1} in (3.8) by Σ_k^{-1}, but in any step we will introduce an additional factor $\alpha_k, k=0,\dots,m-1$. The numbers $\alpha_0,\dots,\alpha_{m-1}$ are positive numbers. Their choice will be discussed later.

$$\begin{array}{rrcl} & \Sigma_0^{-1}\equiv\Sigma_0^{-1}(\alpha_0) &=& \alpha_0\Sigma^{-1}, \\ & E_k &=& \Sigma_k^{-*}\tilde{F}_k \\ (4.28) & \Sigma_{k+1}^{-1}\equiv\Sigma_{k+1}^{-1}(\alpha_{k+1}) &=& \alpha_{k+1}(I+E_k\Delta_{c,k}^{-1}\tilde{F}_k^*)\Sigma_k^{-1},\ k=0,\dots,m-1, \\ & \text{where } \Delta_{c,k} &=& I-[E_k^*\tilde{F}_k+\tilde{F}_k^*E_k-E_k^*S_kE_k],\ k=0,\dots,m-1. \end{array}$$

Obviously $E_k=\Sigma_k^{-*}\tilde{F}_k, \Delta_{c,k}$ can be obtained analogously to Algorithm 3.12,3.15.

The main difference between (4.28) and (3.8) is the definition of $\Delta_{c,k}$. If $\Sigma_k^{-1}=S_k^{-1}$, then $\Delta_{c,k}=S_{c,k}$. The reason for this definition will be given in Corollary 4.30.

Following the definition of $\Sigma_k, k=0,\dots,m$ we define symmetric positive definite operators Δ_k as approximations to S_k.

$$(4.29)\qquad \Delta_0=\Delta,\ \Delta_{k+1}^{-1}:=\Delta^{-1}+\sum_{l=0}^{k}E_l\Delta_{c,l}^{-1}E_l^*,\ k=0,\dots,m-1.$$

If $\alpha_l=1$, for all $l=0,\dots,k$, then by the definition we have $\Delta_k^{-1}-\Sigma_k^{-1}=\Delta^{-1}-\Sigma^{-1}$. In theory, we have $A=S_m-F_mF_m^*$. But if we replace S by Δ,Σ we do not have the low rank property any more. But one can still use Δ_m as preconditioner for A. For S_m we know, that $AS_m^{-1}F_m=F_mS_{c,m}$. So essentially $S_{c,m}$ will correspond to the preconditioned system AS_m^{-1}. Here we have to examine, which influence the use of Σ^{-1} instead of S^{-1} will have for $A\Delta_m^{-1}$. To get estimates we can apply Theorem 4.11.

Corollary 4.30 *Using the notation of (4.28)–(4.29) let* Γ,γ *be constants such that* $\Gamma\geqslant 1\geqslant\gamma$ *and*

$$\gamma\Delta\leqslant S\leqslant\Gamma\Delta$$

for some symmetric positive definite matrix Δ. *Set*

$$\mu_k:=\rho(\tilde{S}_{c,k}^{-1}\tilde{F}_k^*(S_k^{-1}-\Sigma_k^{-1})S_k(S_k^{-1}-\Sigma_k^{-*})\tilde{F}_k),\ k=0,\dots,m-1.$$

Then the following estimates hold:

$$(4.31)\qquad \gamma\lambda_{\min}(S_{c,m})\prod_{k=0}^{m-1}\left(1-\sqrt{\frac{\mu_k}{1+\mu_k}}\right)\Delta_m\leqslant A\leqslant\Gamma\prod_{k=0}^{m-1}\left(1+\sqrt{\frac{\mu_k}{1+\mu_k}}\right)\Delta_m.$$

Furthermore

$$(4.32)\qquad \frac{\lambda_{\max}(\Delta_m^{-1}A)}{\lambda_{\min}(\Delta_m^{-1}A)}\leqslant\frac{\Gamma}{\gamma\lambda_{\min}(S_{c,m})}\prod_{k=0}^{m-1}\left(\sqrt{\mu_k}+\sqrt{1+\mu_k}\right)^2,$$

where

$$(4.33)\qquad \sqrt{\frac{\mu_k}{1+\mu_k}}=\cos\angle(\operatorname{span}\begin{pmatrix}I\\O\end{pmatrix},\ \operatorname{span}\begin{pmatrix}-\Sigma_k^{-*}\tilde{F}_k\\I\end{pmatrix}),\ k=0,\dots,m-1,$$

and the angles are taken with respect to the inner product defined by $\begin{pmatrix} S_k & \tilde{F}_k \\ \tilde{F}_k & I \end{pmatrix}$, $k = 1, \ldots, m$.

Proof:
The proof consists of two steps. First of all we know, that

$$\lambda_{\min}(S_{c,m}) S_m \leqslant A \leqslant S_m \tag{4.34}$$

The right inequaltiy follows directly from the definition of S_m. The left inequality is a consequence of the fact that $\Lambda(S_{c,m}) \cup \{1\} = \Lambda(S_m^{-1} A)$.
Second we have to estimate the relations between Δ_m and S_m in the sense of quadratic forms. To show this we will apply Theorem 4.11 to $\begin{pmatrix} S_k & \tilde{F}_k \\ \tilde{F}_k^* & I \end{pmatrix}$ and the block ILU decomposition

$$\underbrace{\begin{pmatrix} I & 0 \\ \tilde{F}_k^* \Sigma_k^{-1} & I \end{pmatrix}}_{L_k} \underbrace{\begin{pmatrix} S_k & 0 \\ 0 & \Delta_{c,k} \end{pmatrix}}_{D_k} \underbrace{\begin{pmatrix} I & \Sigma_k^{-*} \tilde{F}_k \\ 0 & I \end{pmatrix}}_{L_k^*}.$$

From Theorem 4.11 it follows that

$$\left(1 - \sqrt{\frac{\mu_k}{1+\mu_k}}\right) L_k D_k L_k^* \leqslant \begin{pmatrix} S_k & \tilde{F}_k \\ \tilde{F}_k^* & I \end{pmatrix} \leqslant \left(1 + \sqrt{\frac{\mu_k}{1+\mu_k}}\right) L_k D_k L_k^*,$$

where

$$\sqrt{\frac{\mu_k}{1+\mu_k}} = \cos \angle(\text{span} \begin{pmatrix} I \\ 0 \end{pmatrix}, \text{ span} \begin{pmatrix} \Sigma_k^{-*} \tilde{F}_k \\ I \end{pmatrix})$$

and the angle is taken with respect to the inner product defined by $\begin{pmatrix} S_k & \tilde{F}_k \\ \tilde{F}_k^* & I \end{pmatrix}$. We compare the upper left blocks of $\begin{pmatrix} S_k & \tilde{F}_k \\ \tilde{F}_k^* & I \end{pmatrix}^{-1}$ and $(L_k D_k L_k^*)^{-1}$ and obtain

$$\frac{1}{1 - \sqrt{\frac{\mu_k}{1+\mu_k}}} (S_k^{-1} + \Sigma_k^{-1} \tilde{F}_k \Delta_{c,k}^{-1} \tilde{F}_k^* \Sigma_k^{-*}) \geqslant S_{k+1}^{-1} \geqslant \frac{1}{1 + \sqrt{\frac{\mu_k}{1+\mu_k}}} (S_k^{-1} + \Sigma_k^{-1} \tilde{F}_k \Delta_{c,k}^{-1} \tilde{F}_k^* \Sigma_k^{-*}).$$

This gives us a recursive characterization of S_{k+1}^{-1} and the perturbed system $S_k^{-1} + \Sigma_k^{-1} \tilde{F}_k \Delta_{c,k}^{-1} \tilde{F}_k^* \Sigma_k^{-*}$. Successively applied we will find that

$$S_m^{-1} \geqslant \frac{1}{\prod_{k=0}^{m-1} \left(1 + \sqrt{\frac{\mu_k}{1+\mu_k}}\right)} (S^{-1} + \sum_{k=0}^{m-1} \Sigma_k^{-1} \tilde{F}_k \Delta_{c,k}^{-1} \tilde{F}_k^* \Sigma_k^{-*})$$

and

$$\frac{1}{\prod_{k=0}^{m-1} \left(1 - \sqrt{\frac{\mu_k}{1+\mu_k}}\right)} (S^{-1} + \sum_{k=0}^{m-1} \Sigma_k^{-1} \tilde{F}_k \Delta_{c,k}^{-1} \tilde{F}_k^* \Sigma_k^{-*}) \geqslant S_m^{-1}.$$

Replacing S^{-1} by Δ shows that

$$\gamma \prod_{k=0}^{m-1}\left(1-\sqrt{\frac{\mu_k}{1+\mu_k}}\right)\Delta_m \leqslant S_m \leqslant \Gamma \prod_{k=0}^{m-1}\left(1+\sqrt{\frac{\mu_k}{1+\mu_k}}\right)\Delta_m,$$

which yields the assertion. □

By Corollary 4.30 we get a criterion for the choice of $\alpha_0,\ldots,\alpha_{m-1}$. We should try to choose $\alpha_k > 0$ such that

$$\tilde{\mu}_k(\alpha_k) = \min_{\alpha} \rho(\tilde{S}_{c,k}^{-1}\tilde{F}_k^*(S_k^{-1}-\Sigma_k^{-1}(\alpha))S_k(S_k^{-1}-\Sigma_k^{-*}(\alpha))\tilde{F}_k),\ k=1,\ldots,m.$$

For the simplest case $s_k = 1$ we obtain the scalar problem

$$\tilde{\mu}_k(\alpha_k) = \min_{\alpha} \frac{\tilde{F}_k^*(S_k^{-1}-\alpha\Sigma_k^{-1}(1))S_k(S_k^{-1}-\alpha\Sigma_k^{-*}(1))\tilde{F}_k}{\tilde{S}_{c,k}}.$$

This minimum will be attained for

$$\alpha_k = \frac{\mathrm{Real}\,(\tilde{F}_k^*\Sigma_k^{-*}(1)\tilde{F}_k)}{\tilde{F}_k^*\Sigma_k^{-1}(1)S_k\Sigma_k^{-*}(1)\tilde{F}_k}.$$

It is an open problem how α_k should be chosen in practice, when $s_k > 1$. One problem, which we have in determining α_k is that we know neither $\tilde{F}_k^*S_k^{-1}\tilde{F}_k$ nor $\tilde{S}_{c,k}$ in practice. At least the function $f(\alpha) = \rho(\tilde{S}_{c,k}^{-1}\tilde{F}_k^*(S_k^{-1}-\alpha\Sigma_k^{-1}(1))S_k(S_k^{-1}-\alpha\Sigma_k^{-*}(1))\tilde{F}_k)$ is convex. Thus there exists a global minimum.
Another possibility to use Δ_m for a preconditioner is

$$\hat{A}^{-1} := \Delta_m^{-1} + \Sigma_m^{-*}F_mS_{c,m}^{-1}F_m^*\Sigma_m^{-1},$$

where one has to perform an inner iteration for solving a system with $S_{c,m}$. Here it may be too expensive to compute $\Sigma_m^{-*}F_m$ explicitly, since the number of columns of F_m is r_m which is close to r. For the preconditioning properties of $\hat{A}$ in principle we could apply Corollary 4.30. But if we solve a system with $S_{c,m}$ inexactly or using only a few steps of the cg–iteration, then the bounds for the condition number obtained by Corollary 4.30 are probably far too pessimistic, since we can expect that $\hat{A}$ is not worse as preconditioner than Δ_m. Note that for substructuring methods one must solve the Schur–complement at least approximately, since otherwise the preconditioner would be singular. For Divide & Conquer methods this is not necessary, since we do not need to solve the artificial extended system but only the subsystem corresponding to the original problem.

Beside the nested use of the divide & conquer method based on (1.4) we have another interpretation in the sense of subspace correction methods using the equivalent approach (1.9). For this we assume in addition that $\operatorname{rank} F = r$ and that $H \in \mathrm{M}(r\times n,\mathbb{F})$ satisfies $HF = I$. (1.9) can equivalently be written as

$$O = (I - S^{-1}FT_c^{-1}HA)(I - S^{-1}A), \text{ where } T_c = HAS^{-1}F. \tag{4.35}$$

In order to have an interpretation in the sense of Theorem 4.15, we define $H = (F^*S^{-1}F)^{-1}F^*S^{-1}$ obtain $T_c = (F^*S^{-1}F)^{-1}((S^{-1}F)^T AS^{-1}F)$ and

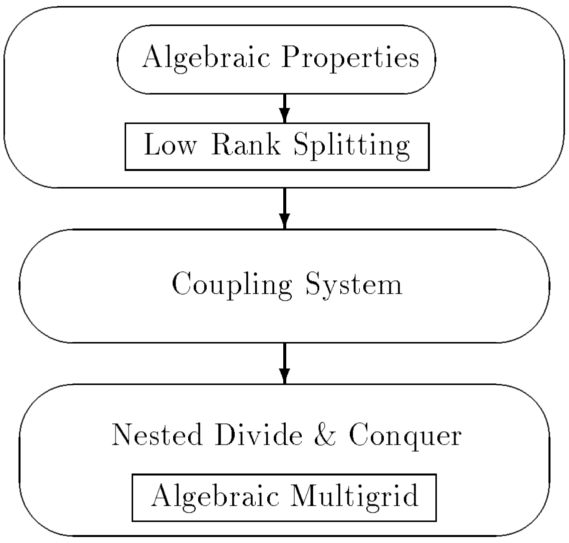
$$O = (I - S^{-1}F\,((S^{-1}F)^T AS^{-1}F)^{-1}\,S^{-1}F^T A)(I - S^{-1}A). \tag{4.36}$$

When replacing S^{-1} by a perturbed matrix Σ^{-1} we obtain a 2–level scheme.
The difference between algebraic multigrid methods in [72] and the algebraic domain decomposition is, that for algebraic domain decomposition the coarse grid corresponds to an invariant subspace of AS^{-1}, while for algebraic multigrid methods the coarse grids are constructed with respect to the relations between the coefficients of the matrix A.

Summary

In this chapter we have shown the close relations between the nested application of both versions of the Sherman–Morrison–Woodbury formula and algebraic multigrid methods. The first version, defined by the nested sequence in (3.4)–(3.8) can be seen as block LU–decomposition of a suitably extended system. When the exact inverse S^{-1} is replaced by an approximate inverse, results for incomplete block LU–decompositions were applicable. The equivalent approach from (1.9) gives us another interpretation of divide & conquer methods as subspace correction method for A. When the exact inverse S^{-1} is replaced by some approximation, we immediately obtain an algebraic multilevel scheme.

From our list of questions on page 10 we have made a suggestion to answer question 1–3. The concept so far can be applied to any low rank splitting and does not require a special splitting $A = S - W$. This can be summarized in the following picture.



To handle the nested divide & conquer method in parallel, i.e., to give an answer to question 4 on page 10, a concrete class of splittings, namely (modified) block diagonal splittings will be considered and a corresponding parallel model will be derived. Therefore modified block diagonal splittings will be discussed in the next chapter.

Chapter 5

Block Jacobi–like Splittings

In this chapter, we will discuss the class of block diagonal splittings. These are the classical block Jacobi splitting as well as modified block Jacobi splittings chosen with respect to minimize the rank of the remaining matrix.
The reason for examining this class of splittings is its simple use for parallel computations. Since we are interested in applying the Sherman–Morrison–Woodbury formula (1.4) to a splitting $A = S - W$ with a nonsingular block diagonal matrix S, we have to discuss in addition factorizations of $W = FG$. F, G should be low rank matrices in order to reduce the size of the coupling system $S_c = I - GS^{-1}F$ from (1.4).

Let $A \in \mathrm{GL}\,(n, \mathbb{F})$, p some positive number counting the number of blocks. Assume that A is partitioned as

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1p} \\ \vdots & & \vdots \\ A_{p1} & \cdots & A_{pp} \end{pmatrix} \tag{5.1}$$

with square diagonal blocks of size $n_1, \ldots, n_p$.

Since we are interested in parallel computations on distributed memory machines with p processors, we have to describe which block of A should be related to which processor. Here we assume that any block A_{qr} is located on processor q and on processor r at the same time. For $q \neq r$, A_{qr} will be stored twice. For sparse matrices this overhead will be acceptable, since many A_{qr} are zero or only have a few number of entries. This memory distribution and its consequences will be the topic of Chapter 7. Here we will only keep in mind this distribution in order to construct special block diagonal splittings. This is of great importance for the factorization of the remaining part.

5.1 General Construction of Block Jacobi–like Splittings

We will start with the classical block Jacobi splitting.

Definition 5.2 *The splitting $A = S_J - W_J$ is called* **<u>block Jacobi splitting</u>**, *if*

$$S_J = \begin{pmatrix} A_{11} & & \mathrm{O} \\ & \ddots & \\ \mathrm{O} & & A_{pp} \end{pmatrix}. \tag{5.3}$$

We are interested in modified block Jacobi splittings $A = S - W$ for large sparse matrices. In addition we would like to have W of low rank and W should should be factorized as $W = FG$. Since we consider block diagonal splittings for reasons of parallel computation, the factorization $W = FG$ should be performed in parallel. This reduces the possibility of efficiently constructing factors F, G with as small rank as possible.

Let I_n be the identity matrix of order n and write I_n as

$$I_n = (E_1, \ldots, E_p)\,, \tag{5.4}$$

where the partitioning corresponds to the block partitioning of A. Then we can write A as

$$\begin{aligned} A &= \underbrace{\sum_{q=1}^{p} E_q A_{qq} E_q^T}_{S_J} - \underbrace{\sum_{1 \leqslant q < r \leqslant p} [E_q,\ E_r] \begin{pmatrix} \mathrm{O} & -A_{qr} \\ -A_{rq} & \mathrm{O} \end{pmatrix} \begin{bmatrix} E_q^T \\ E_r^T \end{bmatrix}}_{W_J} \\ &= S_J - W_J \end{aligned}$$

Note that many pairs $\{q, r\}, q < r$ only exist formally, if A_{qr} and A_{rq} are zero. Therefore we define the index set

$$\mathfrak{I} := \{\{q, r\} :\ q \neq r,\ A_{qr} \neq \mathrm{O} \text{ or } A_{rq} \neq \mathrm{O}\}\,. \tag{5.5}$$

We can restrict ourselves to pairs $\{q, r\} \in \mathfrak{I}$ instead of considering any pair $\{q, r\}$, $1 \leqslant q, r \leqslant p$. Assume that the indices $\mathbf{i}$ of $\mathfrak{I}$ are taken in some fixed order $\mathbf{i}_1, \ldots, \mathbf{i}_s$. Whenever we consider elements $\mathbf{i}$ of $\mathfrak{I}$ as indices for blocks of matrices, we will assume that they are taken in this order.

If a block A_{qr} is stored on processors q and r, then we can find a straightforward factorization if we just construct a factorization of

$$\begin{pmatrix} \mathrm{O} & -A_{qr} \\ -A_{rq} & \mathrm{O} \end{pmatrix}.$$

We can also modify this 2×2 block matrix in the block diagonal positions. Then we obtain a splitting

$$\begin{aligned} A &= \underbrace{\sum_{q=1}^{p} E_q (A_{qq} + \sum_{r:\,\mathbf{i}=\{q,r\} \in \mathfrak{I}} A_{rr}^{\mathbf{i}}) E_q^T}_{S} - \underbrace{\sum_{\mathbf{i}=\{q,r\} \in \mathfrak{I}} [E_q,\ E_r] \begin{pmatrix} A_{qq}^{\mathbf{i}} & -A_{qr} \\ -A_{rq} & A_{rr}^{\mathbf{i}} \end{pmatrix} \begin{bmatrix} E_q^T \\ E_r^T \end{bmatrix}}_{W} \\ &= S - W \end{aligned} \tag{5.6}$$

Definition 5.7 *The splitting $A = S - W$ from (5.6) is called* **modified block Jacobi splitting**.

Note that the block Jacobi splitting from Definition 5.2 is a special case of a modified block diagonal splitting using $A_{rr}^{\{q,r\}} = O$ for any $\{q,r\} \in \mathfrak{I}$.

Assume that $\mathbf{i} = \{q,r\}$ and that $\begin{pmatrix} A_{qq}^{\mathbf{i}} & -A_{qr} \\ -A_{rq} & A_{rr}^{\mathbf{i}} \end{pmatrix}$ is factorized as

$$(5.8) \qquad \begin{pmatrix} A_{qq}^{\mathbf{i}} & -A_{qr} \\ -A_{rq} & A_{rr}^{\mathbf{i}} \end{pmatrix} = \begin{pmatrix} F_q^{\mathbf{i}} \\ F_r^{\mathbf{i}} \end{pmatrix} \begin{pmatrix} G_q^{\mathbf{i}} & G_r^{\mathbf{i}} \end{pmatrix}$$

for suitable matrices $F_q^{\mathbf{i}}, (G_q^{\mathbf{i}})^T \in \mathrm{M}(n_q \times n^{\mathbf{i}}, \mathbb{F})$, $F_r^{\mathbf{i}}, (G_r^{\mathbf{i}})^T \in \mathrm{M}(n_r \times n^{\mathbf{i}}, \mathbb{F})$ for some positive number $n^{\mathbf{i}}$.

We define

$$(5.9) \qquad n_c = \sum_{\mathbf{i} \in \mathfrak{I}} n^{\mathbf{i}}.$$

The ordering $\mathbf{i}_1, \ldots, \mathbf{i}_t$ of the elements in $\mathfrak{I}$ and the corresponding sizes $n^{\mathbf{i}_1}, \ldots, n^{\mathbf{i}_t}$ induce a block partitioning for vectors in $\mathbb{F}^{n_c}$ and matrices in $\mathrm{M}(n \times n_c, \mathbb{F})$, $\mathrm{M}(n_c \times n, \mathbb{F})$. Partition the identity matrix I_{n_c} of size $n_c \times n_c$ columnwise as

$$(5.10) \qquad I_{n_c} = \left(E^{\mathbf{i}_1}, \ldots, E^{\mathbf{i}_t}\right),$$

where $E^{\mathbf{i}_l}$ denotes $n^{\mathbf{i}_l}$ unit vectors one after another.
Here we would like to briefly comment on the notation. Whenever a matrix contains an index of the form $\mathbf{i}$ which is superposed, then the matrix corresponds to what we will later define as the coupling system and its related block size. For all indices q, r which are used as sub indices with a matrix, the matrix corresponds to the initial system and the initial block partitioning. Equation (5.8) is no contradiction to this notation. A matrix with a lower index and an upper index at same time will be used for both systems, the (small) coupling system and the (big) initial system.

Assumption (5.8) is no restriction, since we can always obtain such a factorization, if e.g. $\begin{pmatrix} F_q^{\mathbf{i}} \\ F_r^{\mathbf{i}} \end{pmatrix}$ is set to the identity or if we perform an LU-decomposition of $\begin{pmatrix} A_{qq}^{\mathbf{i}} & -A_{qr} \\ -A_{rq} & A_{rr}^{\mathbf{i}} \end{pmatrix}$.

Example 5.11 *Let*

$$A = \begin{pmatrix} A_{11} & A_{12} & & & & O \\ A_{21} & A_{22} & A_{23} & & & \\ & A_{32} & A_{33} & A_{34} & & \\ & & A_{43} & A_{44} & A_{45} & \\ & & & A_{54} & A_{55} & A_{56} \\ O & & & & A_{65} & A_{66} \end{pmatrix}, S = \left(\begin{array}{cc|cc|cc} A_{11} & A_{12} & & & & \\ A_{21} & A_{22} & & & & \\ \hline & & A_{33} & A_{34} & & \\ & & A_{43} & A_{44} & & \\ \hline & & & & A_{55} & A_{56} \\ & & & & A_{65} & A_{66} \end{array}\right),$$

then a simple way to factorize the remaining matrix $W = S - A$ will be

$$W = \left(\begin{array}{cc||cc} I & & & \\ \hline & I & & \\ & & I & \\ \hline & & & I \end{array}\right)\left(\begin{array}{c|c|c} & -A_{23} & \\ -A_{32} & & \\ \hline\hline & & -A_{45} \\ & -A_{54} & \end{array}\right).$$

Using the (local) factorization (5.8) we obtain a straightforward factorization of W,

$$W = \sum_{\mathbf{i}=\{q,r\}\in\mathfrak{I}} \left(E_q F_q^{\mathbf{i}} + E_r F_r^{\mathbf{i}}\right)\left(G_q^{\mathbf{i}} E_q^T + G_r^{\mathbf{i}} E_r^T\right).$$

In order to recover the four components $F_q^{\mathbf{i}}, F_r^{\mathbf{i}}, G_q^{\mathbf{i}}$ and $G_r^{\mathbf{i}}$ we will introduce four matrices $L, M, \bar{L}, \bar{M}$. We will distinguish between $F_q^{\mathbf{i}}$ and $F_r^{\mathbf{i}}$ ($G_q^{\mathbf{i}}$ and $G_r^{\mathbf{i}}$) by taking the minimum and the maximum value of $\{q, r\}$. Set

$$\begin{aligned} L &= \sum_{\mathbf{i}\in\mathfrak{I}} E_{\min\mathbf{i}} F_{\min\mathbf{i}}^{\mathbf{i}} (E^{\mathbf{i}})^T = \left[\cdots\ E_{\min\mathbf{i}} F_{\min\mathbf{i}}^{\mathbf{i}}\ \cdots\right]_{\mathbf{i}\in\mathfrak{I}} \\ M &= \sum_{\mathbf{i}\in\mathfrak{I}} E_{\max\mathbf{i}} F_{\max\mathbf{i}}^{\mathbf{i}} (E^{\mathbf{i}})^T = \left[\cdots\ E_{\max\mathbf{i}} F_{\max\mathbf{i}}^{\mathbf{i}}\ \cdots\right]_{\mathbf{i}\in\mathfrak{I}} \\ \bar{L} &= \sum_{\mathbf{i}\in\mathfrak{I}} E^{\mathbf{i}} G_{\min\mathbf{i}}^{\mathbf{i}} E_{\min\mathbf{i}}^T = \left[\begin{array}{c} \vdots \\ G_{\min\mathbf{i}}^{\mathbf{i}} E_{\min\mathbf{i}}^T \\ \vdots \end{array}\right]_{\mathbf{i}\in\mathfrak{I}} \\ \bar{M} &= \sum_{\mathbf{i}\in\mathfrak{I}} E^{\mathbf{i}} G_{\max\mathbf{i}}^{\mathbf{i}} E_{\max\mathbf{i}}^T = \left[\begin{array}{c} \vdots \\ G_{\max\mathbf{i}}^{\mathbf{i}} E_{\max\mathbf{i}}^T \\ \vdots \end{array}\right]_{\mathbf{i}\in\mathfrak{I}} \end{aligned} \tag{5.12}$$

By construction we have

$$L\bar{M} = \begin{pmatrix} 0 & -A_{12} & \cdots & -A_{1p} \\ & \ddots & \ddots & \vdots \\ & & \ddots & -A_{p-1,p} \\ 0 & & & 0 \end{pmatrix}.$$

Using the matrices $L, M, \bar{L}, \bar{M}$ we can factorize W as

$$\begin{aligned} W &= \underbrace{\left[\cdots\ E_{\min\mathbf{i}} F_{\min\mathbf{i}}^{\mathbf{i}} + E_{\max\mathbf{i}} F_{\max\mathbf{i}}^{\mathbf{i}}\ \cdots\right]_{\mathbf{i}\in\mathfrak{I}}}_{L+M} \underbrace{\left[\begin{array}{c} \vdots \\ G_{\min\mathbf{i}}^{\mathbf{i}} E_{\min\mathbf{i}}^T + G_{\max\mathbf{i}}^{\mathbf{i}} E_{\max\mathbf{i}}^T \\ \vdots \end{array}\right]_{\mathbf{i}\in\mathfrak{I}}}_{\bar{L}+\bar{M}} \\ &= \underbrace{(L+M)}_{F}\underbrace{(\bar{L}+\bar{M})}_{G} \\ &= FG, \end{aligned} \tag{5.13}$$

where blocks in F, G are taken with respect to the underlying ordering $\mathbf{i}_1, \ldots, \mathbf{i}_s$ of $\mathfrak{I}$. Using this notation we can rewrite the block Jacobi splitting as

$$A = S_J - (\underbrace{L\bar{M} + M\bar{L}}_{W_J}) \tag{5.14}$$

and the modified block Jacobi splitting can be rewritten as

$$A = \underbrace{(S_J + L\bar{L} + M\bar{M})}_{S} - \underbrace{(L + M)(\bar{L} + \bar{M})}_{W} \tag{5.15}$$

with block diagonal matrices $L\bar{L}, M\bar{M}$.
The introduction of F, G and $L, \bar{L}, M, \bar{M}$ as well defines a block partitioning of the form $S_c = (S_c^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{I}}$ for matrices like the coupling system $S_c = I - GS^{-1}F$. According to our convention the indices are superposed and the order of the blocks is assumed to be some ordering $\mathbf{i}_1, \ldots, \mathbf{i}_s$. In the sequel we will use this notation also for other matrices which correspond in size and their partitioning to S_c.

Example 5.16 *Let us assume that A is block tridiagonal. In this case we get $\mathfrak{I} = \{\{1,2\},\{2,3\},\ldots,\{p-1,p\}\}$ and*

$$L = \begin{pmatrix} F_1^{\{1,2\}} & & \\ & \ddots & \\ & & F_{p-1}^{\{p-1,p\}} \\ & & \mathrm{O} \end{pmatrix}, \quad M = \begin{pmatrix} \mathrm{O} & & \\ F_2^{\{1,2\}} & & \\ & \ddots & \\ & & F_p^{\{p-1,p\}} \end{pmatrix}$$

$$\bar{L} = \begin{pmatrix} G_1^{\{1,2\}} & & & \\ & \ddots & & \\ & & G_{p-1}^{\{p-1,p\}} & \mathrm{O} \end{pmatrix}, \quad \bar{M} = \begin{pmatrix} \mathrm{O} & G_2^{\{1,2\}} & & \\ & & \ddots & \\ & & & G_p^{\{p-1,p\}} \end{pmatrix}.$$

$$L\bar{L} = \begin{pmatrix} F_1^{\{1,2\}}G_1^{\{1,2\}} & & & \\ & \ddots & & \\ & & F_{p-1}^{\{p-1,p\}}G_{p-1}^{\{p-1,p\}} & \\ & & & \mathrm{O} \end{pmatrix},$$

$$M\bar{M} = \begin{pmatrix} \mathrm{O} & & & \\ & F_2^{\{1,2\}}G_2^{\{1,2\}} & & \\ & & \ddots & \\ & & & F_p^{\{p-1,p\}}G_p^{\{p-1,p\}} \end{pmatrix}.$$

5.2 Block Jacobi–like Splittings With Respect to Special Classes of Matrices

In this section we are interested in conditions for the 2×2 subproblem (5.8) which ensure that properties of A are inherited by S from a modified block Jacobi splitting and the related coupling system $S_c = I - GS^{-1}F$. E.g. if A is symmetric, S and S_c should also be symmetric. We will do this for the class of symmetric matrices and M–matrices. Essentially we can derive conditions for the 2×2 subproblem (5.8) by applying the theory of Chapter 2.

Corollary 5.17 *Let $A \in \mathrm{GL}\,(n, \mathbb{F})$, $A = A^*$, positive definite. Assume that for any $\mathbf{i} = \{q, r\} \in \mathfrak{I}$, $-A_{qr}, -A_{rq}$ are factored as $-A_{qr} = F_q^{\mathbf{i}} G_r^{\mathbf{i}}, -A_{rq} = F_r^{\mathbf{i}} G_q^{\mathbf{i}} \equiv (G_r^{\mathbf{i}})^*(F_q^{\mathbf{i}})^*$. Define $A_{qq}^{\mathbf{i}}, A_{rr}^{\mathbf{i}}$ by $A_{qq}^{\mathbf{i}} := F_q^{\mathbf{i}}(F_q^{\mathbf{i}})^*$, $A_{rr}^{\mathbf{i}} := (G_r^{\mathbf{i}})^*(G_r^{\mathbf{i}})$.*
Then we already have a factorization

$$\begin{pmatrix} A_{qq}^{\mathbf{i}} & -A_{qr} \\ -A_{rq} & A_{rr}^{\mathbf{i}} \end{pmatrix} = \begin{pmatrix} F_q^{\mathbf{i}} \\ (G_r^{\mathbf{i}})^* \end{pmatrix} \begin{pmatrix} (F_q^{\mathbf{i}})^* & G_r^{\mathbf{i}} \end{pmatrix}.$$

$L, M, \bar{L}, \bar{M}$ from (5.12) and F, G from (5.13) satisfy

$$\bar{L} = L^*, \bar{M} = M^*, \; G = F^*.$$

We obtain a modified block Jacobi splitting of the form $A = S - FF^*$, where S and $S_c = I - F^* S^{-1} F$ are symmetric (Hermitian) positive definite.

Proof:
By construction $A = S - FF^*$ is a modified block Jacobi splitting. S is also symmetric (Hermitian) positive definite, since $S = A + FF^*$. By Lemma 2.8 we obtain that S_c is positive definite. □

By Corollary 5.17 the choice of $A_{qq}^{\mathbf{i}}, A_{rr}^{\mathbf{i}}$ not only reduces the size of the coupling system by a factor of two, it also ensures the symmetry and the positive definiteness of S_c.

Corollary 5.18 *If A is an M–Matrix and $A_{qq}^{\mathbf{i}}, A_{rr}^{\mathbf{i}}$ are diagonal matrices with nonnegative entries for any $\mathbf{i} = \{q, r\} \in \mathfrak{I}$, then S is an M–matrix.*
Assume in addition that we have a factorization

$$\begin{pmatrix} A_{qq}^{\mathbf{i}} & -A_{qr} \\ -A_{rq} & A_{rr}^{\mathbf{i}} \end{pmatrix} = \begin{pmatrix} F_q^{\mathbf{i}} \\ F_r^{\mathbf{i}} \end{pmatrix} \begin{pmatrix} G_q^{\mathbf{i}} & G_r^{\mathbf{i}} \end{pmatrix}$$

for any $\mathbf{i} = \{q, r\} \in \mathfrak{I}$ with nonnegative matrices $F_q^{\mathbf{i}}, F_r^{\mathbf{i}}, G_q^{\mathbf{i}}, G_r^{\mathbf{i}}$, then we obtain a modified block Jacobi splitting of the form $A = S - FG$, where S and $S_c = I - GS^{-1}F$ are M–matrices.

Proof:
Again we get by construction a modified block Jacobi splitting. Using Lemma 2.13 we obtain that S, S_c are M–matrices. $\square$

If A is an M–matrix, then we do not have as much freedom in the choice of $A^{\mathbf{i}}_{qq}, A^{\mathbf{i}}_{rr}$ as in the positive definite case, since we have to care about the sign pattern of S. Nonnegative diagonal matrices $A^{\mathbf{i}}_{qq}, A^{\mathbf{i}}_{rr}$ will in general not allow to reduce the size of S_c efficiently.
Since $\begin{pmatrix} A^{\mathbf{i}}_{qq} & -A_{qr} \\ -A_{rq} & A^{\mathbf{i}}_{rr} \end{pmatrix}$ is a nonnegative matrix we can perform a trivial factorization of this matrix with exclusively nonnegative factors, if we just choose one of the factors to be the identity.

In general it is not necessarily true that F or G has full rank, if $\begin{pmatrix} F^{\mathbf{i}}_q \\ F^{\mathbf{i}}_r \end{pmatrix}$, $\begin{pmatrix} G^{\mathbf{i}}_q & G^{\mathbf{i}}_r \end{pmatrix}$ have full rank. At least for block tridiagonal matrices this is sufficient.

Lemma 5.19 *Assume that A is block tridiagonal.*

If $\left.\begin{array}{c} F_q^{\{q,q+1\}} \\ F_{q+1}^{\{q,q+1\}} \\ G_q^{\{q,q+1\}} \\ G_{q+1}^{\{q,q+1\}} \end{array}\right\}$ *has full rank for all $q = 1, \ldots, p-1$, then* $\left\{\begin{array}{c} L \\ M \\ \bar{L} \\ \bar{M} \end{array}\right.$ *has full rank.*
If L has full rank or if M has full rank, then F has full rank.
If $\bar{L}$ has full rank or if $\bar{M}$ has full rank, then G has full rank.

Proof:
For block tridiagonal matrices $L, M, \bar{L}, \bar{M}$ are block diagonal.
We have $F = L + M$, where F is block bidiagonal. The upper diagonal blocks are those of L, the lower diagonal blocks are those of M. Analogous one can proceed for G. $\square$

Summary

In this chapter we have discussed block Jacobi-like splittings. We have generalized block Jacobi splittings in such a way that the remaining part can be factorized in a straightforward way while modifications for the diagonal blocks are still possible. For positive definite matrices and M–matrices, we have examined which modifications are allowed or necessary to inherit the structure for the modified block diagonal matrix and the related coupling system.
Note that in principle the theory of block Jacobi–like splittings can be extended to the case of overlapping diagonal blocks. This would be rather technical and will be done in a future paper.
The modifications that have been subject of this chapter have addressed the low rank property and modifications have been chosen to inherit structures. The topic of the next chapter will be, how the freedom in constructing low rank splittings can be used to improve the properties of the coupling system beyond just preserving structures.

Chapter 6

Modified Block Jacobi Splittings

In general the main problem in using block Jacobi splittings is that the diagonal blocks may be singular or at least ill–conditioned. Modifications should not only preserve structures like in the positive definite case. They should also improve the condition number of the diagonal blocks or at least improve the properties of the coupling system S_c. Since $(AS^{-1})\,F = F\,S_c$ an improvement of the eigenvalue distribution of S_c could also be useful for the properties of S.

6.1 Motivation for the Choice of Modifications

We will now discuss more general factorizations for modified block Jacobi splittings. Recall that by (5.8) we have local block 2×2 problems

$$\begin{pmatrix} A^{\mathbf{i}}_{qq} & -A_{qr} \\ -A_{rq} & A^{\mathbf{i}}_{rr} \end{pmatrix} = \begin{pmatrix} F^{\mathbf{i}}_q G^{\mathbf{i}}_q & F^{\mathbf{i}}_q G^{\mathbf{i}}_r \\ F^{\mathbf{i}}_r G^{\mathbf{i}}_q & F^{\mathbf{i}}_r G^{\mathbf{i}}_r \end{pmatrix} = \begin{pmatrix} F^{\mathbf{i}}_q \\ F^{\mathbf{i}}_r \end{pmatrix} \begin{pmatrix} G^{\mathbf{i}}_q & G^{\mathbf{i}}_r \end{pmatrix},$$

for any $\mathbf{i} = \{q,r\} \in \mathfrak{I}$. By introducing a nonsingular matrix $X^{\mathbf{i}}$ we can change this block 2×2 problem to

$$\begin{pmatrix} \hat{A}^{\mathbf{i}}_{qq} & -A_{qr} \\ -A_{rq} & \hat{A}^{\mathbf{i}}_{rr} \end{pmatrix} = \begin{pmatrix} F^{\mathbf{i}}_q X^{\mathbf{i}} G^{\mathbf{i}}_q & F^{\mathbf{i}}_q G^{\mathbf{i}}_r \\ F^{\mathbf{i}}_r G^{\mathbf{i}}_q & F^{\mathbf{i}}_r (X^{\mathbf{i}})^{-1} G^{\mathbf{i}}_r \end{pmatrix}. \tag{6.1}$$

By modifying this local problem the block diagonal matrix S will also change. In addition, the modification here does not essentially change the rank, since

$$\begin{pmatrix} F^{\mathbf{i}}_q X^{\mathbf{i}} G^{\mathbf{i}}_q & F^{\mathbf{i}}_q G^{\mathbf{i}}_r \\ F^{\mathbf{i}}_r G^{\mathbf{i}}_q & F^{\mathbf{i}}_r (X^{\mathbf{i}})^{-1} G^{\mathbf{i}}_r \end{pmatrix} = \begin{pmatrix} F^{\mathbf{i}}_q X^{\mathbf{i}} \\ F^{\mathbf{i}}_r \end{pmatrix} (X^{\mathbf{i}})^{-1} \begin{pmatrix} X^{\mathbf{i}} G^{\mathbf{i}}_q & G^{\mathbf{i}}_r \end{pmatrix}. \tag{6.2}$$

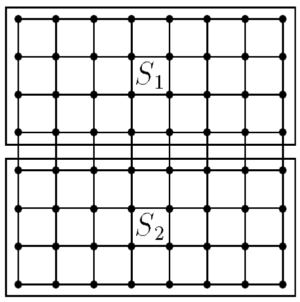
Example 6.3 *Consider as concrete example the matrix A from Example 2.15. A arises from the discretization of the problem*

$$\begin{aligned} -\Delta u &= f \text{ in } [0,1]^2 \\ u &= g \text{ on } \partial[0,1]^2 \end{aligned}$$

using five point star difference discretization [48]. Set

$$S = \left(\begin{array}{c|c} S_1 & \mathrm{O} \\ \hline \mathrm{O} & S_2 \end{array}\right) = \left(\begin{array}{cccc|cccc} T & -I & & & & & & \\ -I & \ddots & \ddots & & & & & \\ & \ddots & T & -I & & & & \\ & & -I & T+I & \mathrm{O} & & & \\ \hline & & & \mathrm{O} & T+I & -I & & \\ & & & & -I & T & \ddots & \\ & & & & & \ddots & \ddots & -I \\ & & & & & & -I & T \end{array}\right)$$

This block matrix S can be read as dividing the discrete problem into two subproblems with a suitable boundary condition on the interface between both subdomains.



The block 2×2 problem here changes from $\begin{pmatrix} I & I \\ I & I \end{pmatrix}$ to $\begin{pmatrix} X & I \\ I & X^{-1} \end{pmatrix}$. Analogously the lower right diagonal block of S_1 and the upper left block of S_2 changes from $T + I$ to $T + X, T + X^{-1}$. These changes only affect those nodes which are located at the internal boundary between both subproblems.

The example has illustrated that the introduction of $X^{\mathbf{i}}$ in (6.1) can be read as the introduction of some kind of algebraic boundary conditions to improve the properties of S, S_c with the additional requirement to maintain the low rank.

If one would like to have two factors instead of three factors on the right hand side of (6.2), then write $X^{\mathbf{i}}$ as $X^{\mathbf{i}} = N^{\mathbf{i}}\bar{N}^{\mathbf{i}}$ and one obtains

$$\begin{pmatrix} F_q^{\mathbf{i}} X^{\mathbf{i}} G_q^{\mathbf{i}} & F_q^{\mathbf{i}} G_r^{\mathbf{i}} \\ F_r^{\mathbf{i}} G_q^{\mathbf{i}} & F_r^{\mathbf{i}} (X^{\mathbf{i}})^{-1} G_r^{\mathbf{i}} \end{pmatrix} = \begin{pmatrix} F_q^{\mathbf{i}} N^{\mathbf{i}} \\ F_r^{\mathbf{i}} (\bar{N}^{\mathbf{i}})^{-1} \end{pmatrix} \begin{pmatrix} \bar{N}^{\mathbf{i}} G_q^{\mathbf{i}} & (N^{\mathbf{i}})^{-1} G_r^{\mathbf{i}} \end{pmatrix}. \tag{6.4}$$

After this introduction we will now formally describe the problem (6.1) as problem for the modified block Jacobi splitting. Instead of writing

$$FG = (L + M)(\bar{L} + \bar{M})$$

we will use the freedom in this factorization and change FG to

$$FG = (LN + M\bar{N}^{-1})(\bar{N}\bar{L} + N^{-1}\bar{M}) \tag{6.5}$$

for some nonsingular $N, \bar{N}$. (6.5) corresponds to (6.4) when taking all $\mathbf{i} \in \mathfrak{I}$. These new F, G from (6.5) differ from the choice of F, G in (5.13) of Chapter 5 if $N, \bar{N}$ are different from the identity . Thus S from (5.15) will change to $S = S_J + LN\bar{N}\bar{L} + M\bar{N}^{-1}N^{-1}\bar{M}$. In order to preserve the block diagonal property of S we have to restrict ourselves to block diagonal matrices $N, \bar{N}$, if $p > 2$.

To do this we now choose nonsingular matrices $N^{\mathbf{i}}, \bar{N}^{\mathbf{i}} \in \mathrm{GL}\,(n^{\mathbf{i}}, \mathbb{F})$ and let

$$(6.6) \qquad \begin{aligned} N &= \sum_{\mathbf{i}\in\mathfrak{I}} E^{\mathbf{i}} N^{\mathbf{i}} (E^{\mathbf{i}})^T = \text{ diag } \left[\cdots\ N^{\mathbf{i}}\ \cdots\right]_{\mathbf{i}\in\mathfrak{I}} \\ \bar{N} &= \sum_{\mathbf{i}\in\mathfrak{I}} E^{\mathbf{i}} \bar{N}^{\mathbf{i}} (E^{\mathbf{i}})^T = \text{ diag } \left[\cdots\ \bar{N}^{\mathbf{i}}\ \cdots\right]_{\mathbf{i}\in\mathfrak{I}} \\ X &= N\bar{N}. \end{aligned}$$

By construction we have

$$(6.7) \qquad F = LN + \bar{M}\bar{N}^{-1}, G = \bar{N}\bar{L} + N^{-1}M, S = LX\bar{L} + MX^{-1}\bar{M}.$$

Hence A can be written as

$$(6.8) \qquad \begin{aligned} A &= S_J - L\bar{M} - M\bar{L} \\ &= \underbrace{(S_J + LX\bar{L} + MX^{-1}\bar{M})}_{S} - \underbrace{(LX + M)X^{-1}(X\bar{L} + \bar{M})}_{FG}, \end{aligned}$$

where $S \equiv S(X)$ is also block diagonal.

Using this formal description for the modified diagonal blocks we are interested in improving the properties of S_c, where

$$(6.9) \qquad S_c = I - (\bar{N}\bar{L} + N^{-1}\bar{M})S^{-1}(LN + M\bar{N}^{-1})$$

or equivalently

$$(6.10) \qquad S_c^{-1} = I + (\bar{N}\bar{L} + N^{-1}\bar{M})A^{-1}(LN + M\bar{N}^{-1}).$$

By multiplying with N from the left and $\bar{N}$ from the right, we obtain

$$(6.11) \qquad NS_c^{-1}\bar{N} = X + (X\bar{L} + \bar{M})A^{-1}(LX + M).$$

Formally it is easier to consider $NS_c^{-1}\bar{N}$ than to consider $NS_c\bar{N}$, since S_c contains the expression $S(X)^{-1}$.

To improve the properties of S_c^{-1} we will investigate the expression

$$(6.12) \qquad \boxed{R(X) = N(S_c^{-1} - \alpha I)\bar{N}}$$

for some $\alpha \in \mathbb{R} \setminus \{0\}$. Ideally we would like to have $N(S_c^{-1} - \alpha I)\bar{N} = O$. In this case S_c^{-1} is a multiple of the identity, which would be optimal. An equation of the form

$$(6.13) R(X) \equiv \bar{M}A^{-1}M + (1-\alpha)X + X\,(\bar{L}A^{-1}M) + (\bar{M}A^{-1}L)\,X + X\,(\bar{L}A^{-1}L)\,X = O$$

is called algebraic Riccati–equation. For the study of Riccati–equations we refer to [58], [54]. To get $N, \bar{N}$ we can derive solutions for X from $R(X)$ and after that we can reconstruct $N, \bar{N}$, e.g., by using the LU–decomposition for X.

So the problem of improving the properties of S_c^{-1} has been traced back to finding block diagonal solutions of algebraic Riccati equations.
In order to have a block diagonal matrix S we need a block diagonal matrix X, if $p > 2$. In general block diagonal solutions of $R(X) = \mathrm{O}$ not necessarily exist. But what we can do is to derive approximate solutions of $R(X) = \mathrm{O}$ and to use these approximate solutions to obtain a resulting block diagonal matrix X. A special case where we will show that explicit solutions exist is the case when the block graph of A is block 2–cyclic (see Definition 6.16). It is an open problem how the theory can be generalized to the non block 2–cyclic case. Of course we could ignore the property 'block 2–cyclic' and choose some approximate X and hope for the best but this seems not to be senseful.

In what follows we will examine $R(X)$ in more detail.

First in Lemma 6.19 we will simplify $R(X)$ for the case that the block graph of the initial matrix A is block 2–cyclic. Under some assumptions we will show that $R(X)$ can be rewritten in the form

$$R(X) = Q + (X + B)C(X + B). \tag{6.14}$$

This will be the topic of Lemma 6.36.
Since the rank of C will be essential for deriving solutions of $R(X) = \mathrm{O}$, we will give sufficient conditions on the nonsingularity of C.
After rewriting $R(X)$ in the form (6.14) we will show in Theorem 6.46 that the equation $R(X) = \mathrm{O}$ has explicit solutions, if α is suitably chosen.
Since explicit solutions of $R(X) = \mathrm{O}$ require the computation of a matrix square root, we will show how this can be avoided.
For symmetric positive matrices we will derive almost optimal modifications in the sense of quadratic forms. Moreover we will discuss the sharpness of these bounds.
The results will be summarized in an algorithm.

Example 6.15 *To illustrate the steps in deriving X we will consider a model matrix*

$$A_n = \left(\begin{array}{cc|cc|cc|cc}
-1 & -1 & 1 & 0 & & & \multicolumn{2}{c}{\multirow{2}{*}{\mathrm{O}}} \\
0 & -1 & 0 & 1 & & & & \\
\hline
-1 & 0 & 2 & -1 & \multicolumn{2}{c|}{\ddots} & & \\
0 & -1 & 0 & 0 & & & & \\
\hline
 & & \multicolumn{2}{c|}{\ddots} & \multicolumn{2}{c|}{\ddots} & 1 & 0 \\
 & & & & & & 0 & 1 \\
\hline
\multicolumn{2}{c|}{\multirow{2}{*}{\mathrm{O}}} & & & -1 & 0 & 2 & -1 \\
 & & & & 0 & -1 & 0 & 0
\end{array}\right) - 10^{-3} I \in \mathrm{M}\,(n \times n, \mathbb{R})$$

for which the steps will be illustrated. Our most frequently choice will be $n = 12$ and $p = 3$.

In this case a quite easy minimum rank splitting will be

$$A_{12} = \left(\begin{array}{cc|cc|cc} A_2 & I & & & & \\ -I & B_2+I & & & & \\ \hline & & B_2-I & I & & \\ & & -I & B_2-I & & \\ \hline & & & & B_2+I & I \\ & & & & -I & B_2 \end{array}\right) - \left(\begin{array}{c|c} O & \\ I & \\ \hline I & \\ & -I \\ \hline & I \\ & O \end{array}\right) \left(\begin{array}{cc|cc|cc} O & I & -I & & & \\ \hline & & & I & I & O \end{array}\right).$$

Here $B_2 = \begin{pmatrix} 2-10^{-3} & -1 \\ 0 & -10^{-3} \end{pmatrix}$ *and* $A_2 = \begin{pmatrix} -1-10^{-3} & -1 \\ 0 & -1-10^{-3} \end{pmatrix}$. *From the splitting* $A = S - FG$ *we obtain the following coupling system* $S_c = I - GS^{-1}F$ *of (1.4):*

$$S_c \approx \left(\begin{array}{cc|cc} 1 & 9.99\cdot 10^2 & 0.501 & -5\cdot 10^{-4} \\ 0 & 10^6 & 0 & 0.5 \\ \hline -0.5 & 5\cdot 10^{-4} & 1.21 & -0.357 \\ 0 & -0.5 & 0 & 0.501 \end{array}\right).$$

Unfortunately this coupling system is very ill–conditioned. Its eigenvalues will be approximately $1.11 \pm 0.489i$, 10^6, 0.501 *and its singular values are approximately* 10^6, 1.37, 1.11, 0.481.
Another choice of a minimum rank splitting will be $A = S - FG$, *where*

$$F = \left(\begin{array}{cc|cc} \multicolumn{2}{c|}{O} & & \\ 1 & 0 & & \\ 0 & 10^{-2} & & \\ \hline 0.5 & -1.5\cdot 10^{-3} & \multicolumn{2}{c}{O} \\ 0 & -10^{-2} & & \\ \multicolumn{2}{c|}{O} & -0.25 & 0.15 \\ & & 0 & 10^{-3} \\ \hline & & 1 & 0 \\ & & 0 & 1 \\ & & \multicolumn{2}{c}{O} \end{array}\right),\ G = \left(\begin{array}{ccc|cccc|ccc} O & 2 & -0.3 & -1 & 0 & \multicolumn{2}{c|}{O} & & & \\ & 0 & -10^2 & 0 & -10^2 & & & & & \\ \hline & & & \multicolumn{2}{c}{O} & 1 & 0 & 4 & -6\cdot 10^2 & O \\ & & & & & 0 & 1 & 0 & -10^3 & \end{array}\right)$$

and $S = A + FG$. *Here we have*

$$S_c \approx \left(\begin{array}{cc|cc} 0.575 & 1.13\cdot 10^{-3} & 6.9\cdot 10^{-2} & -4.21\cdot 10^{-2} \\ 0 & 0.5 & 0 & -0.1 \\ \hline -0.138 & 3.93\cdot 10^{-3} & 0.488 & -2.33\cdot 10^{-2} \\ 0 & 10^{-2} & 0 & 0.499 \end{array}\right)$$

with eigenvalues $0.531 \pm 0.0874i$, $0.5 \pm 0.0316i$ *and singular values* 0.6, 0.55, 0.487, 0.453.
The second minimum rank splitting is far away from being obvious. But it drastically improves the resulting coupling system.

6.2 Simplifications on $R(X)$ for Block 2–Cyclic Matrices

The expression $R(X)$ as well as $NS_c^{-1}\bar{N}$ contain the term A^{-1}, which we will not have in practice. From this point of view a more convenient representation of $R(X)$ is desirable. We will now show that for the class of block 2–cyclic matrices we can extremely simplify $NS_c^{-1}\bar{N}$.

Definition 6.16 *A matrix of the form*

$$\left(\begin{array}{ccc|ccc} A_{1,1} & & \mathrm{O} & A_{1,s+1} & \cdots & A_{1,p} \\ & \ddots & & \vdots & & \vdots \\ \mathrm{O} & & A_{s,s} & A_{s,s+1} & \cdots & A_{s,p} \\ \hline A_{s+1,1} & \cdots & A_{s+1,s} & A_{s+1,s+1} & & \mathrm{O} \\ \vdots & & \vdots & & \ddots & \\ A_{p,1} & \cdots & A_{p,s} & \mathrm{O} & & A_{p,p} \end{array}\right)$$

is called block 2–cyclic.

Many sparse matrices fulfil this condition up to a suitable permutation if the corresponding block graph is bipartite [13],p.4, i.e., up to a suitable relabelling of the blocks. E.g. for block tridiagonal matrices, block circulant matrices (if p is even) firstly the blocks with the odd numbers have to be taken and then the even ones. Similarly matrices whose block graph is a checker board can be reordered.

Lemma 6.17 *Let $A \in \mathrm{GL}\,(n, \mathbb{F})$ with partitioning from (5.1). Assume that A is block 2–cyclic and that the diagonal blocks of A are invertible. If $A = S_J - L\bar{M} - M\bar{L}$ is the standard block Jacobi splitting from (5.14), then $\bar{L}S_J^{-1}M = \mathrm{O}, \bar{M}S_J^{-1}L = \mathrm{O}$.*

Proof:
We will only show $\bar{L}S_J^{-1}M = \mathrm{O}$, the proof of $\bar{M}S_J^{-1}L = \mathrm{O}$ is analogous. By definition we have

$$\bar{L} = \begin{bmatrix} \vdots \\ G^{\mathbf{i}}_{\min \mathbf{i}} E^T_{\min \mathbf{i}} \\ \vdots \end{bmatrix}_{\mathbf{i} \in \mathfrak{I}}, \; M = \begin{bmatrix} \cdots & E_{\max \mathbf{i}} F^{\mathbf{i}}_{\max \mathbf{i}} & \cdots \end{bmatrix}_{\mathbf{i} \in \mathfrak{I}}.$$

Since A is assumed to be block 2–cyclic, there exists a fixed number s such that for any $\mathbf{i} \in \mathfrak{I}$ we always have $\min \mathbf{i} \leqslant s < \max \mathbf{i}$.

Let $\mathbf{i}, \mathbf{j} \in \mathfrak{I}$, then the block of $\bar{L}S_J^{-1}M$ at position $\mathbf{i}, \mathbf{j}$ will be

$$G^{\mathbf{i}}_{\min \mathbf{i}} E^T_{\min \mathbf{i}} S_J^{-1} E_{\max \mathbf{j}} F^{\mathbf{j}}_{\max \mathbf{j}}.$$

Since S_J is block diagonal, this block can only be different from O if $\min \mathbf{i} = \max \mathbf{j}$. Since $\min \mathbf{i} \leqslant s < \max \mathbf{j}$ this case does not occur. □

Example 6.18 *Let us assume that A is block tridiagonal and that p is even (for simplicity). Instead of permuting A we can achieve the same effect by separating odd and even numbers, i.e., we obtain the following factors $L, M, \bar{L}, \bar{M}$:*

$$\begin{aligned} L &= \left[E_1 F_1^{\{1,2\}},\ E_3 F_3^{\{2,3\}},\ E_3 F_3^{\{3,4\}}, \dots, E_{p-1} F_{p-1}^{\{p-2,p-1\}},\ E_{p-1} F_{p-1}^{\{p-1,p\}}\right] \\ M &= \left[E_2 F_2^{\{1,2\}},\ E_2 F_2^{\{2,3\}}, \dots, E_{p-2} F_{p-2}^{\{p-3,p-2\}},\ E_{p-2} F_{p-2}^{\{p-2,p-1\}},\ E_p F_p^{\{p-1,p\}}\right] \end{aligned}$$

$$\bar{L} = \begin{bmatrix} G_1^{\{1,2\}} E_1^T \\ G_3^{\{2,3\}} E_3^T \\ G_3^{\{3,4\}} E_3^T \\ \vdots \\ G_{p-1}^{\{p-2,p-1\}} E_{p-1}^T \\ G_{p-1}^{\{p-1,p\}} E_{p-1}^T \end{bmatrix}, \quad \bar{M} = \begin{bmatrix} G_2^{\{1,2\}} E_2^T \\ G_2^{\{2,3\}} E_2^T \\ \vdots \\ G_{p-2}^{\{p-3,p-2\}} E_{p-2}^T \\ G_{p-2}^{\{p-2,p-1\}} E_{p-2}^T \\ G_p^{\{p-1,p\}} E_p^T \end{bmatrix}.$$

We denote the elements of $L, \bar{L}$ by '$*$' and the elements of $M, \bar{M}$ by '$+$' and illustrate $F = L + M, G = \bar{L} + \bar{M}$ in the following pattern.

$$F = \begin{pmatrix} * & & & & \\ + & + & & & \\ & * & * & & \\ & & + & + & \\ & & & * & * \\ & & & & + \end{pmatrix}, \quad G = \begin{pmatrix} * & + & & & & \\ & + & * & & & \\ & & * & + & & \\ & & & + & * & \\ & & & & * & + \end{pmatrix}.$$

By construction we obtain $A = S_J - L\bar{M} - M\bar{L}$. We illustrate this in the following picture

$$\underbrace{\begin{pmatrix} * & * & & & & \\ * & * & * & & & \\ & * & * & * & & \\ & & * & * & * & \\ & & & * & * & * \\ & & & & * & * \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} * & & & & & \\ & * & & & & \\ & & * & & & \\ & & & * & & \\ & & & & * & \\ & & & & & * \end{pmatrix}}_{S_J} - \underbrace{\begin{pmatrix} & * & & & & \\ & & & & & \\ & * & & * & & \\ & & & & & \\ & & & * & & * \\ & & & & & \end{pmatrix}}_{L\bar{M}} - \underbrace{\begin{pmatrix} & & & & & \\ * & & * & & & \\ & & & & & \\ & & * & & * & \\ & & & & & \\ & & & & * & \end{pmatrix}}_{M\bar{L}}$$

In fact we have $\bar{M} S_J^{-1} L = O, \bar{L} S_J^{-1} M = O$. The modified block diagonal matrix will be $S = S_J + LX\bar{L} + MX^{-1}\bar{M}$. By definition, for any <u>odd</u> i we have

$$S_{i,i} = A_{i,i} + F_i^{\{i-1,i\}} X^{\{i-1,i\}} G_i^{\{i-1,i\}} + F_i^{\{i,i+1\}} X^{\{i,i+1\}} G_i^{\{i,i+1\}}$$

and for any <u>even</u> i we have

$$S_{i,i} = A_{i,i} + F_i^{\{i-1,i\}} (X^{\{i-1,i\}})^{-1} G_i^{\{i-1,i\}} + F_i^{\{i,i+1\}} (X^{\{i,i+1\}})^{-1} G_i^{\{i,i+1\}}.$$

(Here we have to set $F_1^{\{0,1\}}, G_1^{\{0,1\}}, F_n^{\{n,n+1\}}, G_n^{\{n,n+1\}}$ to O).

The property $\bar{L}S_J^{-1}M = O, \bar{M}S_J^{-1}L = O$ from Lemma 6.17 will be essential in simplifying $NS_c^{-1}\bar{N}$, which will be done in Lemma 6.19. Note that Lemma 6.19 will not explicitly require that S_J is block diagonal. This will be true for the whole theory in Section 2,4 and 5.

Lemma 6.19 *Let $A \in \mathrm{GL}\,(n, F)$. Let $A = S_J - L\bar{M} - M\bar{L}$ and assume that S_J is invertible. Set $C = \bar{L}A^{-1}L, D = \bar{M}S_J^{-1}M, \bar{D} = \bar{L}S_J^{-1}L$. Suppose that $\bar{M}S_J^{-1}L = O, \bar{L}S_J^{-1}M = O$. Then $I - \bar{D}D$ is nonsingular,*

(6.20)
$$C = (I - \bar{D}D)^{-1}\bar{D}.$$

Moreover, for $X = N\bar{N}$, $NS_c^{-1}\bar{N}$ from (6.11) can be written as

(6.21)
$$NS_c^{-1}\bar{N} = (X + D)C(X + D) + (X + D).$$

Proof:
We have $A = S_J - [L, M]\begin{bmatrix}\bar{M}\\ \bar{L}\end{bmatrix}$ with nonsingular A and S_J. In this case the related coupling system, here denoted by $G_c = I - \begin{bmatrix}\bar{M}\\ \bar{L}\end{bmatrix} S_J^{-1}[L, M]$, is nonsingular and has the form

$$G_c = \begin{pmatrix} I & -D \\ -\bar{D} & I \end{pmatrix}.$$

We have two ways to write G_c^{-1}. The first one is to use the corresponding Schur–complements $I - D\bar{D}$ and $I - \bar{D}D$, which must be nonsingular:

$$G_c^{-1} = \begin{pmatrix} (I - D\bar{D})^{-1} & D(I - \bar{D}D)^{-1} \\ (I - \bar{D}D)^{-1}\bar{D} & (I - \bar{D}D)^{-1} \end{pmatrix}.$$

The second way is using the Sherman–Morrison–Woodbury formula for G_c:

$$G_c^{-1} = I + \begin{bmatrix}\bar{M}\\ \bar{L}\end{bmatrix} A^{-1}[L, M] = \begin{pmatrix} I + \bar{M}A^{-1}L & \bar{C} \\ C & I + \bar{L}A^{-1}M \end{pmatrix},$$

where $\bar{C} = \bar{M}A^{-1}M$. From this it follows that $C = (I - \bar{D}D)^{-1}\bar{D}, \bar{C} = D(I - \bar{D}D)^{-1}$. This shows (6.20). Next we want to simplify

$$\begin{aligned} NS_c^{-1}\bar{N} &= \bar{M}A^{-1}M + X + X\bar{L}A^{-1}M + \bar{M}A^{-1}LX + X\bar{L}A^{-1}LX \\ &= \bar{C} + X + X(\bar{L}A^{-1}M) + (\bar{M}A^{-1})LX + XCX. \end{aligned}$$

In order to obtain (6.21) we have to show that

(6.22)
$$\bar{C} = D + DCD, \bar{L}A^{-1}M = CD, \bar{M}A^{-1}L = DC.$$

In this case a straightforward calculation yields (6.21).

It remains to show (6.22). Calculating $D + DCD$ gives

$$D + DCD = D + D(I - \bar{D}D)^{-1}\bar{D}D = D(I - \bar{D}D)^{-1}(I - \bar{D}D + \bar{D}D) = \bar{C},$$

which shows the equation for $\bar{C}$ in (6.22). Again using the Sherman–Morrison–Woodbury formula, this time for A^{-1}, we find that

$$A^{-1} = (S_J - [L,M]\begin{bmatrix}\bar{M}\\ \bar{L}\end{bmatrix})^{-1} = S_J^{-1} + S_J^{-1}[L,M]\,G_c^{-1}\begin{bmatrix}\bar{M}\\ \bar{L}\end{bmatrix} S_J^{-1}.$$

It follows that

$$\begin{aligned}\bar{M}A^{-1}L &= \bar{M}S_J^{-1}L + \bar{M}S_J^{-1}[L,M]\,G_c^{-1}\begin{bmatrix}\bar{M}\\ \bar{L}\end{bmatrix} S_J^{-1}L\\ &= [0,D]\,G_c^{-1}\begin{bmatrix}0\\ \bar{D}\end{bmatrix}\\ &= D(I-\bar{D}D)^{-1}\bar{D}\\ &= DC.\end{aligned}$$

Analogously we can proceed to obtain $\bar{L}A^{-1}M = CD$. This shows (6.22). $\square$

By Lemma 6.19, $NS_c^{-1}\bar{N}$ no longer explicitly contains A^{-1}. Essentially one only needs $D, \bar{D}$, which only require the block diagonal matrix S_J^{-1}.
If C^{-1} exists, then it is easier to access than C, since $C^{-1} = \bar{D}^{-1} - D$ and $\bar{D}$,D are almost block diagonal, i.e., $\bar{D}$ is block diagonal if the block size is suitably enlarged. $\bar{D} = \bar{L}S_J^{-1}L$ can be written as

$$\bar{D} = \sum_{q=1}^{p} \bar{L}E_q A_{qq}^{-1}E_q^T L,$$

where $E_1,\dots,E_p$ are the block columns of the identity matrix from (5.4). It follows that $\bar{D}$ can be written as a sum of up to p matrices $\bar{L}E_qA_{qq}^{-1}E_q^TL$. By the definition of $L, \bar{L}$ from (5.12), we only have to consider those $q \in \{1,\dots,p\}$ which satisfy $q = \min \mathbf{i}$ for at least one $\mathbf{i} \in \mathfrak{I}$. For q with this property we have

$$\bar{L}E_qA_{qq}^{-1}E_q^TL = \sum_{\mathbf{i},\mathbf{j}\in\mathfrak{I}:\,\mathbf{i}\cap\mathbf{j}=\{q\}} E^{\mathbf{i}}G_q^{\mathbf{i}}A_{qq}^{-1}F_q^{\mathbf{j}}(E^{\mathbf{j}})^T.$$

It immediately follows that for $q \neq r$, the blocks of $\bar{L}E_qA_{qq}^{-1}E_q^TL$ and $\bar{L}E_rA_{rr}^{-1}E_r^TL$ do not intersect. Thus $\bar{D}$ is a matrix having the block entries of $\bar{L}E_qA_{qq}^{-1}E_q^TL$ as diagonal blocks of larger size. Analogous properties hold for D. We will illustrate this in an example.

Example 6.23 *For the block tridiagonal matrix from Example 6.18 we obtain that the matrices $D, \bar{D}$ will be block diagonal with the following block pattern.*

$$\bar{D} = \begin{pmatrix} * & & & & & \\ & * & * & & & \\ & * & * & & & \\ & & & \ddots & & \\ & & & & * & * \\ & & & & * & * \end{pmatrix}, D = \begin{pmatrix} * & * & & & & \\ * & * & & & & \\ & & \ddots & & & \\ & & & * & * & \\ & & & * & * & \\ & & & & & * \end{pmatrix}.$$

If $\bar{D}^{-1}$ exists,then $C^{-1} = \bar{D}^{-1} - D$ is block tridiagonal.
Particularly for the model matrix A_{12} from (6.15) we obtain

$$\bar{D} \approx \left(\begin{array}{cc|cc} 1 & -2\cdot 10^{-3} & \multicolumn{2}{c}{\multirow{2}{*}{O}} \\ 0 & -1 & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{O}} & 0.4 & -0.201 \\ & & 0 & -10^{-3} \end{array}\right), \quad D \approx \left(\begin{array}{cc|cc} -0.4 & 0.201 & -0.2 & -0.4 \\ 0 & 10^{-3} & 0 & -1 \\ \hline 0.2 & 0.4 & -0.4 & 0.201 \\ 0 & 1 & 0 & 10^{-3} \end{array}\right).$$

As a nice consequence of Lemma 6.19 we can see, that the nonsingularity of $\bar{D}$ and C are strongly connected. This can be seen from the following observation. We have

$$\bar{D}x = 0 \Leftrightarrow (I - \bar{D}D)^{-1}\bar{D}x = 0 \Leftrightarrow Cx = 0 \text{ and } y\bar{D} = 0 \Leftrightarrow y\bar{D}(I - D\bar{D})^{-1} = 0 \Leftrightarrow yC = 0.$$

This connection can be generalized to the singular case in Corollary 6.24.

Corollary 6.24 *Under the assumptions of Lemma 6.19 we have the following relation between $\bar{D}$ and C.*
Suppose that $\bar{D} = U \begin{pmatrix} \bar{D}_{11} & O \\ O & O \end{pmatrix} V^*$ for orthogonal (unitary) matrices $U = (U_1, U_2)$, $V = (V_1, V_2)$ and a nonsingular $\bar{D}_{11}$ of order $s \times s$ (this can be achieved, e.g. using the singular value decomposition). Set $\begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix} = V^*DU$. Then C has rank s and

$$C = U_1(I - \bar{D}_{11}D_{11})^{-1}\bar{D}_{11}V_1^*. \tag{6.25}$$

Proof:
From $C = (I - \bar{D}D)^{-1}\bar{D}$ it follows that

$$\begin{aligned} C &= U\left(I - \begin{pmatrix} \bar{D}_{11} & O \\ O & O \end{pmatrix}\begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix}\right)^{-1}\begin{pmatrix} \bar{D}_{11} & O \\ O & O \end{pmatrix}V^* \\ &= U\begin{pmatrix} I - \bar{D}_{11}D_{11} & -\bar{D}_{11}D_{12} \\ O & I \end{pmatrix}^{-1}\begin{pmatrix} \bar{D}_{11} & O \\ O & O \end{pmatrix}V^* \\ &= U_1\left(I - \bar{D}_{11}D_{11}\right)^{-1}\bar{D}_{11}V_1^*. \end{aligned}$$

□

The close relation between $\bar{D}$ and C will give us the opportunity to get the left and right null space of C from $\bar{D}$. But as we have already shown, $\bar{D}$ is almost block diagonal. The nonsingularity of $\bar{D}$ as well as left and right null space can be computed in parallel, since only S_J^{-1} is required.
In the sequel we will restrict ourselves to the case of nonsingular C. The generalization to the singular case is more technical. Several assertions from the nonsingular case still hold in the singular case. In the appendix we will discuss an example for the singular case.

In this section we have simplified the expression $R(X)$ from (6.12) for the class of block 2–cyclic matrices which has lead to a more convenient representation of $R(X)$ that does not longer contain A^{-1} explicitly. For the representation of $NS_c^{-1}\bar{N}$ in (6.21) the rank of C will be essential. Since we will only consider the nonsingular case here, sufficient conditions on the nonsingularity of C will be presented in the next section.

6.3 Sufficient Conditions for the Nonsingularity of C

In Section 2 we have shown that the problem of improving S_c can be traced back to finding solutions of the algebraic Riccati equation $(X+D)C(X+D)+(X+D)-\alpha X=0$. Since the rank of C will be essential in this equation we will give a sufficient condition for the nonsingularity of $C=\bar{L}A^{-1}L$. A necessary condition will be that $L,\bar{L}$ have full rank. In general this will not be sufficient except for some classes of matrices.

In the sequel we assume that A is block 2–cyclic, i.e., there exists s such that for any $A_{rq}\neq 0, A_{qr}\neq 0$ we have $q\leqslant s<r$. In other words, for any pair $\mathbf{i}=\{q,r\}\in\mathfrak{I}$ we can assume that $q\leqslant s<r$. The blocks of $L,\bar{L}$ are obtained from the factorization of $-A_{qr}=F_q^{\mathbf{i}}G_r^{\mathbf{i}}, -A_{rq}=F_r^{\mathbf{i}}G_q^{\mathbf{i}}$. By construction $F_q^{\mathbf{i}}$ will become a part of L and $G_q^{\mathbf{i}}$ will belong to $\bar{L}$. In (6.1) we did not discuss how this decomposition should be constructed. A very simple factorization could be

$$-A_{qr}=\underbrace{I}_{F_q^{\{q,r\}}}\underbrace{(-A_{qr})}_{G_r^{\{q,r\}}},\ -A_{rq}=\underbrace{(-A_{rq})}_{F_r^{\{q,r\}}}\underbrace{I}_{G_q^{\{q,r\}}}.$$

In this case the block columns of L and block rows of $\bar{L}$ would correspond to the block unit vectors E_q, E_r. This factorization is much too rough, since A_{qr},A_{rq}^T typically have many rows which are zero and we should make use of this property.

Example 6.26 *Consider A from Example 6.18. There we have the following L.*

$$L=\left[E_1F_1^{\{1,2\}},\ E_3F_3^{\{2,3\}},\ E_3F_3^{\{3,4\}},\dots,E_{p-1}F_{p-1}^{\{p-2,p-1\}},\ E_{p-1}F_{p-1}^{\{p-1,p\}}\right].$$

This matrix has the following patterns.

$$L=\left(\begin{array}{ccccc} * & & & & \\ & * & * & & \\ & & & * & * \end{array}\right).$$

Obviously it suffices to require that $[F_{2i-1}^{\{2i-2,2i-1\}},F_{2i-1}^{\{2i-1,2i\}}]$ has full rank to ensure that L has full rank. Analogous arguments can be used for $\bar{L}$.
For the model matrix A_n from (6.15) with $n=12,p=3$ we have already seen in Example 6.18 that $L,\bar{L}$ will have full rank. The situation changes if we choose $p=4$ and choose S_J

as the block diagonal part of A_{12} with diagonal blocks all of size 3×3. We obtain

$$L = \left(\begin{array}{cc|cc|cc} 0 & 0 & & & & \\ 1 & 0 & \multicolumn{2}{c|}{\mathrm{O}} & \multicolumn{2}{c}{\mathrm{O}} \\ 0 & 1 & & & & \\ \hline \multicolumn{2}{c|}{\mathrm{O}} & \multicolumn{2}{c|}{\mathrm{O}} & \multicolumn{2}{c}{\mathrm{O}} \\ \hline & & 1 & 0 & 0 & 0 \\ \multicolumn{2}{c|}{\mathrm{O}} & 0 & 1 & 1 & 0 \\ & & 0 & 0 & 0 & 1 \\ \hline \multicolumn{2}{c|}{\mathrm{O}} & \multicolumn{2}{c|}{\mathrm{O}} & \multicolumn{2}{c}{\mathrm{O}} \end{array}\right), \quad \bar{L} = L^T.$$

In this case $L, \bar{L}$ will be singular.

Example 6.26 has illustrated that a full rank requirement on $L, \bar{L}$ is not unrealistic. The entries of A_{qr} correspond to the rows $\sum_{w=1}^{q-1} n_w + 1, \ldots, \sum_{w=1}^{q} n_w$ of A from (5.1), since any diagonal block A_{ww} is assumed to be a $n_w \times n_w$ matrix. Under these assumptions we can find numbers $l_1, \ldots, l_s \in \{\sum_{w=1}^{q-1} n_w + 1, \ldots, \sum_{w=1}^{q} n_w\}$ such that A_{qr} only has nonzero entries in those rows corresponding to $l_1, \ldots, l_s$ and likewise A_{rq} only has nonzero entries only in columns corresponding to $l_1, \ldots, l_s$. We will denote this set by $\mathcal{A}_{qr}$. We can formally define $\mathcal{A}_{qr}$ by

$$(6.27) \qquad \mathcal{A}_{qr} := \{l \in \{1, \ldots, n\} : \; e_l^T E_q A_{qr} \neq \mathrm{O} \text{ or } A_{rq} E_q^T e_l^T \neq \mathrm{O}\}, \; q \leqslant s,$$

where e_l denotes the l–th unit vector in $\mathbb{R}^n$. $\mathcal{A}_{qr}$ is the union of nonzero rows of A_{qr} and A_{rq}^T as part of the initial matrix A.

Using this set $\mathcal{A}_{qr}$ we can find a factorization of A_{qr} for $q \leqslant s$, $\mathbf{i} = \{q, r\}$ such that

$$(6.28) \qquad -A_{qr} = F_q^{\mathbf{i}} G_r^{\mathbf{i}}, \; F_q \in \mathrm{M}\,(n_q \times \#\mathcal{A}_{qr}, \mathbb{R}), \big(e_l^T E_q F_q^{\mathbf{i}}\big)_{l \in \mathcal{A}_{qr}} \in \mathrm{GL}\,(\#\mathcal{A}_{qr}, \mathbb{R}).$$

Here $\#\mathcal{A}_{qr}$ denotes the number of elements. Likewise we are able to achieve

$$(6.29) \qquad -A_{rq} = F_r^{\mathbf{i}} G_q^{\mathbf{i}}, \; G_q \in \mathrm{M}\,(\#\mathcal{A}_{qr} \times n_q, \mathbb{R}), \; \big(G_q^{\mathbf{i}} E_q^T e_l\big)_{l \in \mathcal{A}_{qr}} \in \mathrm{GL}\,(\#\mathcal{A}_{qr}, \mathbb{R}).$$

By construction the factors $F_q^{\mathbf{i}}, G_q^{\mathbf{i}}$ are full rank matrices for any $q \leqslant s$ and $L, \bar{L}$ only consist of $F_q^{\mathbf{i}}, G_q^{\mathbf{i}}$ for $q \leqslant s$! The easiest way to find such $F_q^{\mathbf{i}}, G_q^{\mathbf{i}}$ is just to take a diagonal matrix with respect to the elements of $\mathcal{A}_{qr}$.

Example 6.30 *Consider the model matrix $A = A_n$ from (6.15) and $n = 12, p = 3$. Here $A_{1,2}, A_{2,1}, A_{2,3}, A_{3,2}$ are given by*

$$A_{1,2} = A_{2,3} = \left(\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right), \quad A_{2,1} = A_{3,2} = \left(\begin{array}{cc|cc} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right).$$

The union of nonzero rows of $A_{1,2}, A_{2,1}^T$ will be $3, 4$ and the union of nonzero rows of $A_{2,3}, A_{3,2}^T$ will be $7, 8$ as rows of the whole matrix A. We can factorize $A_{1,2}, A_{2,1}$ as

$$-A_{1,2} = \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \hline -1 & 0 \\ 0 & -1 \end{pmatrix}}_{F_1^{\{1,2\}}} \underbrace{\left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right)}_{G_2^{\{1,2\}}}, \quad -A_{2,1} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \hline 0 & 0 \\ 0 & 0 \end{pmatrix}}_{F_2^{\{1,2\}}} \underbrace{\left(\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right)}_{G_1^{\{1,2\}}}.$$

The roles of $A_{2,3}, A_{3,2}$ have to be interchanged in order satisfy the condition block 2–cyclic for $A_{1,2}$.

$$-A_{2,3} = \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \hline 1 & 0 \\ 0 & 1 \end{pmatrix}}_{F_2^{\{2,3\}}} \underbrace{\left(\begin{array}{cc|cc} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{array}\right)}_{G_3^{\{2,3\}}}, \quad -A_{3,2} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \hline 0 & 0 \\ 0 & 0 \end{pmatrix}}_{F_3^{\{2,3\}}} \underbrace{\left(\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right)}_{G_3^{\{2,3\}}}.$$

The factorization which has been use here ensures that $\begin{pmatrix} F_1^{\{1,2\}} \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} \mathrm{O} \\ F_3^{\{2,3\}} \end{pmatrix}$ have full rank. These matrices will become block columns of L. Likewise $\left(G_1^{\{1,2\}}, \mathrm{O}\right), \left(\mathrm{O}, G_3^{\{2,3\}}\right)$ have full rank and will become block rows of $\bar{L}$.

Using this kind of factorization we have a sufficient condition for the nonsingularity of $L, \bar{L}$ in terms of $\mathcal{A}_{q,r}$, $\{q,r\} \in \mathfrak{I}$.

Lemma 6.31 *Let $A \in \mathrm{GL}(n, \mathbb{F})$ with partitioning from (5.1). Assume that A is block 2–cyclic,i.e., there exists s such that any pair $\{q,r\} \in \mathfrak{I}$ satisfies $q = \min\{q,r\} \leqslant s < r = \max\{q,r\}$. Let $A = S_J - L\bar{M} - M\bar{L}$ be the standard block Jacobi splitting from (5.14) and assume that for $q \leqslant s < r$, A_{qr}, A_{rq} are factored according to (6.28), (6.29). Suppose that for any $q = 1, \dots, s$,*

$$\mathcal{A}_{qr} \cap \mathcal{A}_{q,r'} = \emptyset, \text{ for any } \{q,r\} \neq \{q,r'\} \in \mathfrak{I}. \tag{6.32}$$

Then $L, \bar{L}$ have full rank. Moreover, the matrices

$$(e_l^T L)_{l \in \bigcup\limits_{q \leqslant s < r} \mathcal{A}_{qr}}, \ (\bar{L} e_l)_{l \in \bigcup\limits_{q \leqslant s < r} \mathcal{A}_{qr}} \tag{6.33}$$

are block diagonal and nonsingular. They coincide with the nonzero rows/columns of $L, \bar{L}$.

Proof:
We will only show that L has full rank. The proof for $\bar{L}$ is analogous. Define for $q = 1, \dots, s$, L_q by

$$L_q = \left[\cdots F_q^{\{q,r\}} \cdots\right]_{r:\,\{q,r\} \in \mathfrak{I}}$$

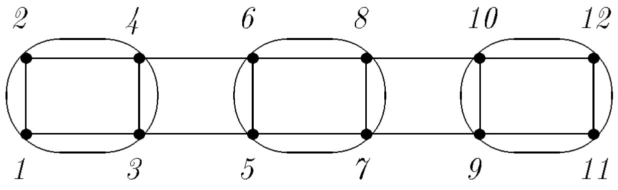
Then L can be written as

$$L = [E_1 L_1, \ldots, E_s L_s] \,.$$

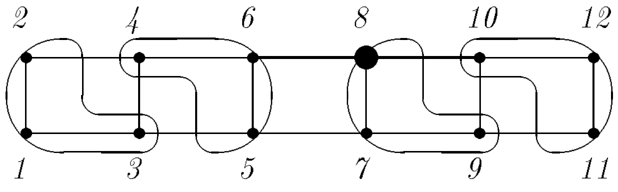
L has full rank, if any L_q has full rank. But since $\mathcal{A}_{q,r} \cap \mathcal{A}_{q,r'} = \emptyset$ for any $r \neq r'$, L_q has full rank if any $F_q^{\{q,r\}}$ has full rank. This is true by construction.
The matrices from (6.33) are just the restrictions of $L, \bar{L}$ to their nonzero rows/columns. Therefore they have to be nonsingular block diagonal matrices. □

The condition $\mathcal{A}_{q,r} \cap \mathcal{A}_{q,r'} = \emptyset$ for any $r \neq r'$ has an interpretation in terms of graph theory. We assign an undirected graph $(\mathcal{V}, \mathcal{E})$ with the matrix A by setting $\mathcal{V} = \{1, \ldots, n\}$, $\mathcal{E} = \{\{i, j\} : a_{ij} \neq 0, i \neq j\}$. For any diagonal block A_{qq} we can analogously define $\mathcal{V}_q$, $\mathcal{E}_q$. The set $\mathcal{A}_{qr}$ describes those nodes of $\mathcal{V}_q$ which have a common edge with some node of $\mathcal{V}_r$, $q \leqslant s < r$. In other words $\mathcal{A}_{qr}$ describes the connection between the subgraphs $(\mathcal{V}_q, \mathcal{E}_q)$ and $(\mathcal{V}_r, \mathcal{E}_r)$ as part of the whole graph. The requirement $\mathcal{A}_{q,r} \cap \mathcal{A}_{q,r'} = \emptyset$ for any $r \neq$ means, that any node of $\mathcal{V}_q$,$q \leqslant s$ has at most one connection to another subgraph.

Example 6.34 *Consider the matrix A_n from (6.15) and $n = 12, p = 3$. The following picture shows the undirected graph of A_n the subgraphs of any diagonal block.*



In order to satisfy the block 2–cyclic requirement, we assume that the diagonal blocks with the odd numbers are taken first. Here we have $\mathcal{A}_{1,2} = \{3, 4\}$, $\mathcal{A}_{3,2} = \{9, 10\}$. Condition (6.32) will be satisfied, since for $q = 1$ or $q = 3$ there only exists one set $\mathcal{A}_{qr}$.
Now we assume that $p = 4$ and the block diagonal part of A will have blocks of size 3×3.



We have $\mathcal{A}_{1,2} = \{2, 3\}$, $\mathcal{A}_{3,2} = \{7, 8\}$, $\mathcal{A}_{3,4} = \{8, 9\}$. Condition (6.31) is violated, since $\mathcal{A}_{3,2} \cap \mathcal{A}_{8,9} = \{8\} \neq \emptyset$.

The assumptions that $L, \bar{L}$ have full rank are in general only necessary to guarantee the nonsingularity of C. But for $L, \bar{L}$ from Lemma 6.31 the situation is better, since these matrices are essentially block diagonal matrices, i.e., to have a nonsingular $C = \bar{L} A^{-1} L$ it suffices to require that the reduced diagonal block of A^{-1} according to $\bigcup_{q \leqslant s < r} \mathcal{A}_{qr}$ is nonsingular.

Corollary 6.35 *Assume that the conditions of Lemma 6.31 are fulfilled. Suppose that any diagonal block of A is nonsingular, i.e., for any subset $I \subseteq \{1,\dots,n\}$ the matrix $(a_{ij})_{i,j\in I}$ is nonsingular. Then C is nonsingular.*
Partition C, C^{-1} as $C = (C^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{J}}$ $C^{-1} = ((C^{-1})^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{J}}$ with respect to (5.10). Then any diagonal block of C, C^{-1} with respect to this partitioning will be nonsingular.

Proof:
Note that if any diagonal block of A is nonsingular, this will be true for any diagonal block of A^{-1} and any Schur–complement of A. We can factorize

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & O \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & O \\ O & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{pmatrix} \begin{pmatrix} I & A_{11}^{-1}A_{12} \\ O & I \end{pmatrix}.$$

The nonsingularity of A together with the nonsingularity of A_{11} imply that $S_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is nonsingular. The block of A^{-1} in the lower right corner will be S_{22}^{-1}. This argument can be used as well for any symmetric permutation of A and any principal submatrix of A.
$C = \bar{L}A^{-1}L$ and since by Lemma 6.31 $\bar{L}, L$ are essentially block diagonal, the reduced matrix

$$(e_l^T A^{-1} e_m)_{l,m \in \bigcup\limits_{q \leqslant s < r} \mathcal{A}_{qr}}$$

must be nonsingular, too. This matrix differs from C up to nonsingular block diagonal factors from the left and from the right side. The partitioning is precisely that of C. So C is nonsingular and the same is true for any diagonal block of C, C^{-1} with respect to the partitioning $C = (C^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{J}}$, $C^{-1} = ((C^{-1})^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{J}}$. □

The assumption that any diagonal block of A is nonsingular will be fulfilled for a wide class of matrices, e.g. diagonally dominant matrices[8],p.186, M–matrices[8],p.133. In principle this assumption can be extended to the case of matrices with an underlying block structure like block diagonally dominant matrices[35], generalized M–matrices [63],[64]. So for a wide class of matrices we can derive the nonsingularity of C from the graph theoretic criterion $\mathcal{A}_{q,r} \cap \mathcal{A}_{q,r'} = \emptyset, r \neq r'$ given in Lemma 6.31.

In this section we have discussed sufficient conditions on A and its graph to guarantee the nonsingularity of C and the diagonal blocks of C, C^{-1}. This shows that the requirement on the nonsingularity of C is not too unrealistic. Moreover, for several classes of matrices we have a graph theoretical criterion that ensures the nonsingularity of C, provided that the off–diagonal blocks A_{qr}, A_{rq} are suitably factorized.
The criterion presented in this section allows us to investigate more precisely the case where C is nonsingular and to find conditions under which the algebraic Riccati–equation $R(X)$ from (6.12) has explicit solutions.

6.4 The General Nonsingular Case

Using the general formulation for modified block Jacobi splittings, which has been introduced in Section 1 we will examine the choice of X more detailed. We will only consider the case when C from (6.20) is nonsingular. In Section 3 we have given a sufficient condition for the nonsingularity of C.

The following lemma plays a key role for the subsequent theory. It will show that under relatively general assumptions, i.e., $\bar{M}S_J^{-1}L = \bar{L}S_J^{-1}M = O$ and a nonsingularity requirement on $\bar{D}$ from Lemma 6.19 we are able to use quadratic expansion. This extremely simplifies the treatment of Riccati equations and will be a step towards finding explicit solutions.

Lemma 6.36 *Let $A \in \mathrm{GL}(n,F)$. Let $A = S_J - L\bar{M} - M\bar{L}$ and assume that S_J is nonsingular. Assume that $\bar{M}S_J^{-1}L = O, \bar{L}S_J^{-1}M = O$ and that $\bar{D}$ from Lemma 6.19 is nonsingular. In this case C from Lemma 6.19 is nonsingular and we can define T_1, T_2 by*

$$T_1(\alpha) = \frac{\alpha-1}{2}C^{-1} - D = \frac{\alpha-1}{2}\bar{D}^{-1} - \frac{\alpha+1}{2}D, \tag{6.37}$$

$$T_2(\alpha) = \left(\frac{\alpha-1}{2}\right)^2 C^{-1} - \alpha D = \left(\frac{\alpha-1}{2}\right)^2 \bar{D}^{-1} - \left(\frac{\alpha+1}{2}\right)^2 D. \tag{6.38}$$

Then we get for $X = N\bar{N}$

$$N(S_c^{-1} - \alpha I)\bar{N} = (X - T_1(\alpha))C(X - T_1(\alpha)) - T_2(\alpha). \tag{6.39}$$

Proof:
By Lemma 6.19 we have

$$\begin{aligned} N(S_c^{-1} - \alpha I)\bar{N} &= (X+D)C(X+D) - (\alpha-1)(X+D) + \alpha D \\ &= (X + D - \frac{\alpha-1}{2}C^{-1})C(X + D - \frac{\alpha-1}{2}C^{-1}) - \left(\frac{\alpha-1}{2}\right)^2 C^{-1} + \alpha D \\ &= (X - T_1(\alpha))C(X - T_1(\alpha)) - T_2(\alpha). \end{aligned}$$

□

By Lemma 6.36 we are able to obtain

$$ZCZ = T_2(\alpha), \tag{6.40}$$

if there exists a matrix square root of $CT_2(\alpha)$. If a square root exists, then we would be able to solve this equation exactly as long as we have any freedom in the choice of $X = Z + T_1(\alpha)$. For the existence of a matrix square root of a matrix B it is sufficient that all eigenvalues of B are in the open right complex plane. The next lemma will prepare this step.

Lemma 6.41 *Assume that $\bar{D}$ from Lemma 6.19 is nonsingular and set $W = \bar{D}D$ (by Lemma 6.19, $I - W$ is nonsingular). Then (6.40) is equivalent to*

$$Z^2 = CT_2(\alpha), \tag{6.42}$$

where $Z = C(X - T_1)$. If $\alpha \in \mathbb{R} \setminus \{0\}$ satisfies

$$(\alpha + 1)^2 \geqslant 4\alpha \operatorname{Re}\mu \tag{6.43}$$

for any eigenvalue μ of $(I - W)^{-1}$, then the eigenvalues of $CT_2(\alpha)$ have nonnegative real parts. If in addition $\alpha \geqslant 1$, then the eigenvalues of $CT_1(\alpha)$ also have nonnegative real parts. If α strictly satisfies inequality (6.43), then the eigenvalues of $CT_1(\alpha), CT_2(\alpha)$ do not touch the imaginary axis. In this case there exists B such that

$$B^2 = CT_2(\alpha) \tag{6.44}$$

and all eigenvalues λ of B satisfy $\operatorname{Re}\lambda > |\operatorname{Im}\lambda|$.

Proof:

Equation (6.42) immediately follows from (6.39).

If μ is an eigenvalue of $(I - W)^{-1}$, then $1 - \frac{1}{\mu}$ is an eigenvalue of W.

We have $CT_2(\alpha) = (I - W)^{-1}((\frac{\alpha-1}{2})^2 I - (\frac{\alpha+1}{2})^2 W)$. Thus the eigenvalues of $CT_2(\alpha)$ are

$$\mu\left(\left(\frac{\alpha-1}{2}\right)^2 - \left(\frac{\alpha+1}{2}\right)^2\left(1 - \frac{1}{\mu}\right)\right) = -\alpha\mu + \frac{(\alpha+1)^2}{4}$$

for any eigenvalue μ of $(I - W)^{-1}$. This shows (6.43). Analogously we can proceed to show that the eigenvalues of $CT_1(\alpha)$ are

$$-\mu + \frac{\alpha+1}{2},$$

which has nonnegative real part, if $\alpha \geqslant 1$ satisfies (6.43).

If in inequality (6.43) the equality is excluded, then no real part of $CT_1(\alpha)$ or $CT_2(\alpha)$ will be 0. In this case the eigenvalues of $CT_2(\alpha)$ are in the open right complex plane and thus a square root B of $CT_2(\alpha)$ exists. The eigenvalues of B are the square roots of the eigenvalues of $CT_2(\alpha)$ and the angle can be chosen such that the real parts dominate the absolute values of the imaginary parts. □

Remark: Condition (6.43) if fulfilled for all sufficiently large α.

Example 6.45 *For A_{12} from (6.15) the eigenvalues of $(I - \bar{D}D)^{-1}$ are*

$$0.999 \pm 0.0315i,\ 0.781 \pm 0.0246i.$$

For this example the matrix square root of $CT_2(\alpha)$ will exist and be real for any α

$$CT_2(\alpha) \approx \left(\frac{\alpha+1}{2}\right)^2 I - \alpha \left(\begin{array}{cc|cc} 0.707 & 0.147 & -0.122 & -0.144 \\ 0 & 0.998 & 0 & 0.998 \\ \hline 4.88 \cdot 10^{-2} & -2.48 \cdot 10^{-2} & 0.854 & 2.41 \cdot 10^{-2} \\ 0 & -9.98 \cdot 10^{-4} & 0 & 0.999 \end{array}\right)$$

Lemma 6.41 has shown that there exists a matrix square root of $CT_2(\alpha)$ if α is suitably chosen. Therefore we have explicit solutions of the algebraic Riccati–equation from (6.13). The next theorem will summarize the results.

Theorem 6.46 *Let $A, S_J \in \mathrm{GL}\,(n, \mathbb{F})$, let $A = S_J - L\bar{M} - M\bar{L}$. Assume that $\bar{D}$ from Lemma 6.19 is nonsingular and $\bar{M}S_J^{-1}L = \mathrm{O}, \bar{L}S_J^{-1}M = \mathrm{O}$. If α satisfies (6.43), then the Riccati equation $N(S_c^{-1} - \alpha I)\bar{N} = \mathrm{O}$ from (6.39) has extremal solutions*

$$X_{\pm}(\alpha) = T_1(\alpha) \pm C^{-1}B \tag{6.47}$$

with B from (6.44) and $X = N\bar{N}$.

Proof:
This is just a summary of the previous results. □

In this section we have shown that solutions of the algebraic Riccati–equation exist if α is suitably chosen and the additional requirement $\bar{L}S_J^{-1}M = \mathrm{O}$, $\bar{M}S_J^{-1}L = \mathrm{O}$ is fulfilled. In practice the explicit solution will be too expensive to compute since it requires the computation of a matrix square root of $CT_2(\alpha)$. In the next section we will show that the computation of a matrix square root can be avoided.

6.5 Approximate Solution $T_1(\alpha)$

The theory so far is only suitable for the existence of solutions of the Riccati–equation $N(S_c^{-1} - \alpha I)\bar{N} \equiv (X - T_1(\alpha))C(X - T_1(\alpha)) - T_2(\alpha) \overset{!}{=} \mathrm{O}$. For the exact solution of $(C(X - T_1(\alpha))^2 - CT_2(\alpha){=}\mathrm{O}$ one has to compute the square root of CT_2, which will be very expensive. So we have have to approximate the theoretical solution. In practice we cannot solve the Riccati–equation exactly, since X is only allowed to be block diagonal.
In this section we will show that for sufficiently large α, $X = T_1(\alpha)$ from (6.37) is almost a solution of $N(S_c^{-1} - \frac{\alpha-1}{2}I)\bar{N} = \mathrm{O}$, i.e. for this choice of X we will have $S_c^{-1} = \frac{\alpha-1}{2}(I + E)$, where E is a perturbation matrix of small norm. The advantage of this result is that $T_1(\alpha)$ is much easier to compute than the exact solution, since it does not require the matrix square root of $CT_2(\alpha)$.

For this we will show that for sufficiently large α, $T_1(\alpha), T_2(\alpha)$ are close to be a multiple of C^{-1}.

Lemma 6.48 *Using the assumptions and notation of Lemma 6.36 we assume that $\bar{D}$ is nonsingular (then by Lemma 6.41, $I-\bar{D}D$ is nonsingular). Let $0<\delta<1$ be fixed. If $\alpha>1$ is chosen such that*

$$(\alpha+1)^2 \geqslant 4\alpha\|(I-\bar{D}D)^{-1}\|/\delta, \tag{6.49}$$

then $T_1(\alpha), T_2(\alpha)$ satisfy

$$T_1(\alpha)=\frac{\alpha+1}{2}C^{-1}(I-E_1),\ T_2(\alpha)=\left(\frac{\alpha+1}{2}\right)^2 C^{-1}(I-E_2), \tag{6.50}$$

where $\|E_1\|, \|E_2\| \leqslant \delta$.

Proof:
By definition we have that $T_2(\alpha) = \left(\frac{\alpha+1}{2}\right)^2 C^{-1} - \alpha\bar{D}^{-1} = \left(\frac{\alpha+1}{2}\right)^2 C^{-1}\left(I-\frac{4\alpha}{(\alpha+1)^2}(I-\bar{D}D)^{-1}\right)$. We set $E_2 = \frac{4\alpha}{(\alpha+1)^2}(I-\bar{D}D)^{-1}$. Using (6.49) we obtain $\|E_2\|\leqslant\delta$. Analogously we can show that $T_1(\alpha)=\frac{\alpha+1}{2}C^{-1}(I-E_1)$ for a matrix E_1 with norm less than or equal to δ. □

Using Lemma 6.48 we will now show that $X=T_1(\alpha)$ almost solves the equation $N(S_c^{-1}-\frac{\alpha-1}{2}I)\bar{N}$. More precisely we will obtain $S_c^{-1}=\frac{\alpha-1}{2}(I+E)$ for a small perturbation E.

Theorem 6.51 *We use the assumptions and notation of Lemma 6.36. Assume that $\bar{D}$ is nonsingular and that α is chosen such that $\hat{\alpha}=\frac{\alpha-1}{2}>1$ satisfies (6.49).*
If $\|\frac{2}{\alpha+1}CX-I\|\leqslant\varepsilon<1$, then we obtain for $X=N, \bar{N}=I$

$$S_c^{-1}=\frac{\alpha-1}{2}(I+E) \tag{6.52}$$

and E satisfies

$$\|E\|\leqslant 2\varepsilon+\frac{\delta\varepsilon+\delta^2/4+3/4\delta}{1-\varepsilon}. \tag{6.53}$$

Particularly for $X=T_1(\alpha)$, E satisfies

$$\|E\|\leqslant\frac{\delta}{1-\delta} \tag{6.54}$$

Proof:
By (6.39) and Lemma 6.48 we have for $X=N, \bar{N}=I$

$$\begin{aligned}(S_c^{-1}-\frac{\alpha-1}{2}I) &= (CX)^{-1}CN(S_c^{-1}-\hat{\alpha}I)\bar{N}\\ &= (CX)^{-1}\left[(CX-CT_1(\hat{\alpha}))^2-CT_2(\hat{\alpha})\right]\\ &= \frac{(\hat{\alpha}+1)^2}{4}(CX)^{-1}\left[(\frac{2}{\hat{\alpha}+1}CX-I+E_1)^2-I+E_2\right]\\ &= \frac{(\alpha+1)^2}{16}(CX)^{-1}\left[(\frac{4}{\alpha+1}CX-I+E_1)^2-I+E_2\right]\end{aligned}$$

We set $\frac{2}{\alpha+1}CX = I + E_x$ and obtain

$$\begin{aligned}
(S_c^{-1} - \frac{\alpha-1}{2}I) &= \frac{\alpha+1}{8}(I+E_x)^{-1}\left[(I+2E_x+E_1)^2 - I + E_2\right] \\
&= \frac{\alpha+1}{8}\left[4E_x + (I+E_x)^{-1}(4E_xE_1 + 4E_1E_x + E_1^2 + 2E_1 + E_2)\right] \\
&=: \frac{\alpha-1}{2}E.
\end{aligned}$$

From $\frac{\alpha-1}{2} > 1$ it follows that

$$\|E\| \leqslant \frac{\alpha+1}{4(\alpha-1)}\left[4\varepsilon + \frac{8\delta\varepsilon + \delta^2 + 3\delta}{1-\varepsilon}\right] \leqslant 2\varepsilon + \frac{4\delta\varepsilon + \delta^2/2 + 3/2\delta}{1-\varepsilon}.$$

Particularly for $X = T_1(\alpha)$ we obtain more precisely

$$(S_c^{-1} - \frac{\alpha-1}{2}I) = \frac{\alpha-1}{2}(CT_1(\alpha))^{-1}C\bar{D}^{-1} =: \frac{\alpha-1}{2}E.$$

Here we have

$$\|E\| \leqslant \frac{2}{(\hat{\alpha}+1)(1-\delta)}\frac{\delta(\hat{\alpha}+1)^2}{4\hat{\alpha}} \leqslant \frac{\delta}{1-\delta}.$$

□

Example 6.55 *For the model matrix A_n from (6.15) and $n = 12, p = 3$ it suffices if α satisfies the inequality*

$$\frac{(\alpha+1)^2}{4\alpha} \geqslant 1.62/\delta,$$

since the smallest singular value of $I - \bar{D}D$ will be $\approx 6.19 \cdot 10^{-1}$. For δ we prescribe the values $1/4, 1/8$. In this case we obtain $\alpha \geqslant 27.8, 53.7$ and

$$T_1(27.8) \approx \left(\begin{array}{cc|cc} 19.2 & -2.92 & 2.88 & 5.76 \\ 0 & -13.4 & 0 & 14.4 \\ \hline -2.88 & -5.76 & 39.3 & -6.72\cdot 10^3 \\ 0 & -14.4 & 0 & -1.34\cdot 10^4 \end{array}\right), \quad T_1(53.7) \approx \left(\begin{array}{cc|cc} 37.3 & -5.54 & 5.47 & 10.9 \\ 0 & -26.4 & 0 & 27.3 \\ \hline -5.47 & -10.9 & 76.8 & -1.32\cdot 10^4 \\ 0 & -27.3 & 0 & -2.63\cdot 10^4 \end{array}\right).$$

Essentially $T_1(53.7) \approx 2T_1(27.8)$. For S_c^{-1} we obtain in this case

$$S_c(T_1(27.8))^{-1} \approx 13.4(I - \underbrace{\left(\begin{array}{cc|cc} -0.0521 & -0.0104 & 0.00379 & 0.0115 \\ 0 & -1.6\cdot 10^{-5} & 0 & 0.08 \\ \hline -0.00784 & 0.00157 & -0.064 & 0.00260 \\ 0 & -0.0694 & 0 & -0.149 \end{array}\right)}_{E(T_1(27.8))}),$$

$$S_c(T_1(53.7))^{-1} \approx 26.4(I - \underbrace{\left(\begin{array}{cc|cc} -0.0268 & 0.00536 & 0.00189 & 0.00570 \\ 0 & -2.47\cdot 10^{-6} & 0 & 0.0393 \\ \hline -0.00394 & 0.000787 & -0.0319 & 0.00126 \\ 0 & -0.0366 & 0 & -0.0758 \end{array}\right)}_{E(T_1(53.7))}).$$

The relative error has essentially been reduced by a factor 2, since $\|E(T_1(27.8))\|_2 \leqslant 0.181$, $\|E(T_1(53.7))\|_2 \leqslant 0.0918$. This shows that $T_1(\alpha)$ is almost a solution of $S_c^{-1} \overset{!}{=} \frac{\alpha-1}{2}I$.

By Theorem 6.51 the problem of finding X has been reduced to approximating $T_1(\alpha)$ or $\frac{\alpha+1}{2}C^{-1}$ by an appropriate nonsingular block diagonal matrix X. So far it is an open problem which choice of a block diagonal matrix should be used. By Theorem 6.51 to minimize the norm $\|\frac{2}{\alpha+1}CX - I\|$ is one theoretical possibility. But in practice we can access only C^{-1} explicitly. Even if C is available a block diagonal solution which minimizes $\|\frac{2}{\alpha+1}CX - I\|$ is known only for a few norms. And also in this case X may be singular. In principle we could choose X as the block diagonal part of $T_1(\alpha)$ or $\frac{\alpha+1}{2}C^{-1}$. But this requires that the diagonal blocks of $T_1(\alpha)$, $\frac{\alpha+1}{2}C^{-1}$ respectively, are nonsingular. But we do not necessarily have this property. In Section 3 we have at least presented a sufficient criterion to ensure that the diagonal blocks of C, C^{-1} are nonsingular.

6.6 Optimal Choice of Modifications in the Symmetric Positive Definite Case

The problem of improving the properties of the coupling system S_c has been traced back to finding approximate solutions of an algebraic Riccati equation. We have shown in Theorem 6.51, that $X = T_1(\alpha)$ and $\frac{\alpha+1}{2}C^{-1}$ almost solve the algebraic Riccati equation $S_c^{-1} - \alpha I \overset{!}{=} O$. In practice this approximate solution still has to be replaced by a block diagonal matrix. Under relatively general conditions it is hard to discuss the quality of an approximate block diagonal choice X. The situation is different, when we consider symmetric positive definite matrices. Again we will concentrate on the case where the block graph of A is 2–cyclic. For this case it has been shown in [30] that the optimal block diagonal preconditioning matrix will be given by the block diagonal part S_J of A itself. Here we examine modified block diagonal splittings and the related matrix $S = LXL^* + MXM^*$ need not necessarily give an optimal preconditioner for the initial matrix A. The reason is that in general $S^{-1}A$ will have 1 as an eigenvalue, while for the coupling system this is typically not the case.

Example 6.56 *As symmetric positive definite model matrix we will take*

$$A_n = \left(\begin{array}{cc|cc|cc|cc}
0.732 & -1 & -1 & 0 & & & \multicolumn{2}{c}{O} \\
-1 & 3 & 0 & -1 & & & & \\
\hline
-1 & 0 & 4 & -1 & \multicolumn{2}{c|}{\ddots} & & \\
0 & -1 & -1 & 4 & & & & \\
\hline
& & \multicolumn{2}{c|}{\ddots} & \multicolumn{2}{c|}{\ddots} & -1 & 0 \\
& & & & & & 0 & -1 \\
\hline
\multicolumn{2}{c|}{O} & & & -1 & 0 & 4 & -1 \\
& & & & 0 & -1 & -1 & 4
\end{array}\right) \in \mathrm{M}(n \times n, \mathbb{R})$$

$n = 12$ and $p = 3$ will be our most frequent choice, i.e., S_J will be the block diagonal part of A with blocks of size 4×4.

Assume that A is positive definite. The problem of finding optimal modifications has already been studied in [11] for block tridiagonal matrices. Here we will use a different

approach.
Corresponding to Lemma 5.17 we assume that $A_{q,r}$ is factorized as $-A_{q,r} = F_q^{\{q,r\}} G_r^{\{q,r\}}$ for all $q < r$ and set

$$\left(\begin{array}{cc} A_{qq}^{\{q,r\}} & -A_{qr} \\ -A_{rq} & A_{rr}^{\{q,r\}} \end{array} \right) := \left(\begin{array}{c} F_q^{\{q,r\}} \\ (G_r^{\{q,r\}})^* \end{array} \right) \left(\begin{array}{cc} (F_q^{\{q,r\}})^* & G_r^{\{q,r\}} \end{array} \right). \tag{6.57}$$

In this case we obtain $\bar{M} = M^*, \bar{L} = L^*$ and it is only natural also to require $\bar{N} := N^*$ in (6.6). Thus we have

$$\begin{aligned} A &= S_J - LM^* - ML^* \\ &= \underbrace{(S_J + LXL^* + MX^{-1}M^*)}_{S} - \underbrace{(LN + MN^{-*})}_{F} \underbrace{(N^*L^* + N^{-1}M^*)}_{F^*}. \end{aligned} \tag{6.58}$$

So S will be symmetric positive definite and the coupling system

$$S_c = I - (LN + MN^{-*})^* S^{-1} (LN + MN^{-*}) \tag{6.59}$$

is also positive definite in this case.

In the sequel we will assume that $\bar{D} = L^* S_J^{-1} L$ is nonsingular. In the symmetric positive definite case we can find simpler conditions which ensure the nonsingularity of $\bar{D}$ and as well the nonsingularity of C. Lemma 6.31 can be simplified to the following corollary.

Corollary 6.60 *Let $A \in \mathrm{GL}\,(n, \mathbb{F})$ with partitioning from (5.1). Let $A = S_J - LM^* - ML^*$ be the standard block Jacobi splitting from (5.14). Assume that A is block 2–cyclic,i.e., there exists s such that any pair $\{q, r\} \in \mathfrak{I}$ can satisfy $q \leqslant s < r$. Assume that in (6.57) the factorization at least $F_q^{\{q,r\}}$ has full rank. Suppose that for all $q = 1, \ldots, s$, $\mathcal{A}_{q,r} \cap \mathcal{A}_{q,r'} = \emptyset$ for any $\{q, r\} \neq \{q, r'\} \in \mathfrak{I}$, where $\mathcal{A}_{q,r}, \mathcal{A}_{q,r'}$ are taken from (6.27). Then L has full rank and $\bar{D}, C$ from Lemma 6.19 are symmetric positive definite.*

By Corollary 6.60 we essentially have a purely graph theoretical criterion to guarantee the positive definiteness of $\bar{D}, C$.

In the positive definite case we will try to minimize $N(S_c^{-1} - \alpha I)N^* = (X - T_1(\alpha))C(X - T_1(\alpha) - T_2(\alpha)$ from (6.39) with $T_1(\alpha), T_2(\alpha)$ from (6.37),(6.38) in the sense of quadratic forms, i.e., we will consider the problem

$$\gamma I \leqslant S_c \leqslant \Gamma I, \tag{6.61}$$

or equivalently

$$\frac{1}{\Gamma} I \leqslant S_c^{-1} \leqslant \frac{1}{\gamma} I, \tag{6.62}$$

for $0 < \gamma \leqslant \Gamma < 1$ such that $\frac{\Gamma}{\gamma} \overset{!}{=} \min$. To solve this problem we will follow the arguments used in [32],[33].

We set

$$R(Z, \alpha) = ZCZ - T_2(\alpha), \quad \text{where } Z = X - T_1(\alpha). \tag{6.63}$$

So the problem has been reduced to finding symmetric (Hermitian) solutions of algebraic Riccati inequalities. We state this result as a Lemma.

Lemma 6.64 *Using the notation of Lemma 6.19 and (6.63),we assume that $\bar{D}$ is nonsingular (in this case C is nonsingular). Then the problem of finding a symmetric (Hermitian) positive $X = NN^*$ such that*

$$S_c \leqslant \Gamma I,$$

is equivalent to finding a symmetric (Hermitian) solution Z of the Riccati inequality

$$R(Z, \frac{1}{\Gamma}) \geqslant O,\ Z = X - T_1(\alpha) \tag{6.65}$$

such that $X > O$.
The problem of finding a symmetric (Hermitian) positive $X = NN^*$ such that

$$S_c \geqslant \gamma I,$$

is equivalent to finding a symmetric (Hermitian) solution Z of the Riccati inequality

$$R(Z, \frac{1}{\gamma}) \leqslant O,\ Z = X - T_1(\alpha). \tag{6.66}$$

such that $X > O$.

The relations between $T_1(\alpha)$ and $T_2(\alpha)$ from (6.37),(6.38) are described in the following lemma.

Lemma 6.67 *Assume that $\bar{D}$ from Lemma 6.19 is nonsingular. Let $\bar{D} = KK^*$, $W = K^*DK$ and assume that λ, Λ satisfy sharply*

$$\lambda I \leqslant (I - W)^{-1} \leqslant \Lambda I. \tag{6.68}$$

Then $\lambda \geqslant 1$ and for $\alpha > 1$ the following inequalities hold:

$$T_2(\alpha) \leqslant O \Longleftrightarrow (\alpha + 1)^2 \leqslant 4\alpha\lambda. \tag{6.69}$$

$$T_2(\alpha) \geqslant O \Longleftrightarrow (\alpha + 1)^2 \geqslant 4\alpha\Lambda. \tag{6.70}$$

In this case we have

$$\frac{\alpha - 1}{2} C^{-1} \geqslant T_1(\alpha) \geqslant \frac{\alpha^2 - 1}{4\alpha} C^{-1} \geqslant O. \tag{6.71}$$

Proof:
$\bar{D}$ is assumed to be nonsingular. By Corollary 6.24 this implies that C is nonsingular and positive definite. From this it follows that $K^*C^{-1}K = K^*\bar{D}^{-1}K - K^*DK = I - W > O$. Since $W \geqslant O$ we have $\lambda \geqslant 1$.
Let μ be an eigenvalue of $(I - W)^{-1}$. Then the eigenvalues of $K^*T_2(\alpha)K$ are

$$\left(\frac{\alpha - 1}{2}\right)^2 I - \left(\frac{\alpha + 1}{2}\right)^2 \left(1 - \frac{1}{\mu}\right) = -\alpha + \frac{(\alpha + 1)^2}{4\mu}.$$

It immediately follows that

$$T_2(\alpha) \leqslant O \iff (\alpha+1)^2 \leqslant 4\alpha\lambda, \ T_2(\alpha) \geqslant O \iff (\alpha+1)^2 \geqslant 4\alpha\Lambda.$$

From $T_2(\alpha) \geqslant O$ it follows that

$$T_1(\alpha) \ \geqslant \ \frac{\alpha-1}{2}C^{-1} - \frac{1}{\alpha}\left(\frac{\alpha-1}{2}\right)^2 C^{-1} = \frac{\alpha^2-1}{4\alpha}C^{-1}$$

which shows the inequality in the middle of (6.71), while the left and the right inequalities of (6.71) are clear. □

The positive semidefiniteness of $T_1(\alpha)$ and $T_2(\alpha)$ can be seen as a special case of Lemma 6.41 where we have shown that for sufficiently large α the eigenvalues of $CT_1(\alpha)$ and $CT_2(\alpha)$ are in the right half plane. Here we have in addition C is positive definite.

By Lemma 6.67 we have to compute solutions of Riccati inequalities, provided they exist. For $R(Z,\alpha) \geqslant O$ it is clear that we always have solutions, while for $R(Z,\alpha) \leqslant O$ we have to require that $T_2(\alpha) \geqslant O$. The following Lemma determines sets of symmetric (Hermitian) solutions of both Riccati inequalities.

Lemma 6.72 *Using the notation of Lemma 6.67 we assume that C is factorized as $C = PP^*$ and that $\alpha, \alpha_1, \alpha_2 > 1$. If $(\alpha+1)^2 \geqslant 4\alpha\Lambda$, then the Riccati equation $R(Z,\alpha) = O$ has two extremal solutions $Z_\pm(\alpha)$,*

(6.73) $$Z_\pm(\alpha) = \pm P^{-*}\left(P^*T_2(\alpha)P\right)^{1/2} P^{-1}.$$

Any symmetric (Hermitian) Z satisfying

(6.74) $$Z \leqslant Z_-(\alpha) \ \vee \ Z_+(\alpha) \leqslant Z.$$

solves the Riccati inequality

$$R(Z,\alpha) \geqslant O.$$

The Riccati inequality

$$R(Z,\alpha) \leqslant O$$

has the set of symmetric (Hermitian) solutions Z satisfying

(6.75) $$Z_-(\alpha) \leqslant Z \leqslant Z_+(\alpha).$$

If $(\alpha+1)^2 < 4\alpha\Lambda$, then the Riccati inequality

$$R(Z,\alpha) \leqslant O$$

has no symmetric (Hermitian) solution Z.
If $(\alpha+1)^2 \leqslant 4\alpha\lambda$, then any symmetric (Hermitian) Z is a solution of the Riccati inequality

$$R(Z,\alpha) \geqslant O.$$

Proof:
(6.73) is just the special case of symmetric positive definite matrices in Theorem 6.46.
(6.74) and (6.75) follow immediately from (6.73).
The Riccati inequality $\mathrm{O} \leqslant ZCZ \leqslant T_2(\alpha)$ can have solutions only if $T_2(\alpha)$ is positive semidefinite. By Lemma 6.67 this is only possible if $(\alpha+1)^2 \geqslant 4\alpha\Lambda$.
For $(\alpha+1)^2 \leqslant 4\alpha\lambda$ the matrix $T_2(\alpha)$ is negative semidefinite, so $R(Z,\alpha) \geqslant \mathrm{O}$ is always true. $\square$

From Lemma 6.72 we can derive solutions X of problem (6.61),(6.62).

Theorem 6.76 *Using the notation of Lemma 6.67 we assume that C is factorized as $C = PP^*$ and $\alpha, \alpha_1, \alpha_2 > 1$. If $(\alpha+1)^2 \geqslant 4\alpha\Lambda$, then the Riccati equation*

$$S_c = \frac{1}{\alpha} I$$

has two extremal symmetric (Hermitian) positive definite solutions $X_-(\alpha), X_+(\alpha)$, defined by

(6.77) $$X_\pm(\alpha) = T_1(\alpha) \pm P^{-*}\left(P^* T_2(\alpha) P\right)^{1/2} P^{-1}.$$

They satisfy

(6.78) $$\left.\begin{array}{ccc} X_+(\alpha_1) & > & X_+(\alpha_2) \\ X_-(\alpha_1) & < & X_-(\alpha_2) \end{array}\right\}, \quad \textit{for any } \alpha_1 > \alpha_2, \quad \textit{such that } (\alpha_2+1)^2 \geqslant 4\alpha_2\Lambda,$$

(6.79) $X_-(\alpha_1) < X_+(\alpha_2)$, *for any α_1, α_2 such that* $(\alpha_1+1)^2 \geqslant 4\alpha_1\Lambda, (\alpha_2+1)^2 \geqslant 4\alpha_2\Lambda$.

Let $1 \geqslant \Gamma \geqslant \gamma$ and assume that Γ satisfies $(1+\frac{1}{\Gamma})^2 \geqslant \frac{4}{\Gamma}\Lambda$. Then any symmetric (Hermitian) positive definite X satisfying

(6.80) $$\mathrm{O} < X \leqslant X_-(\frac{1}{\Gamma}) \ \vee \ X_+(\frac{1}{\Gamma}) \leqslant X$$

solves the Riccati inequality

$$S_c \leqslant \Gamma I.$$

The Riccati inequality

$$S_c(X) \geqslant \gamma I$$

has the nonempty set of symmetric (Hermitian) positive definite solutions X, defined by

(6.81) $$X_-(\frac{1}{\gamma}) \leqslant X \leqslant X_+(\frac{1}{\gamma})$$

Any symmetric (Hermitian) positive definite solutions X satisfying

(6.82) $$X_-(\frac{1}{\gamma}) \leqslant X \leqslant X_-(\frac{1}{\Gamma}) \ \vee \ X_+(\frac{1}{\Gamma}) \leqslant X \leqslant X_+(\frac{1}{\gamma})$$

solves the combined Riccati inequality

$$\gamma I \leqslant S_c(X) \leqslant \Gamma I.$$

S_c is a multiple of the Identity, if $\gamma = \Gamma$ is chosen.
If $(1+\frac{1}{\gamma})^2 < \frac{4}{\gamma}\Lambda$, then the Riccati inequality

$$S_c \geqslant \gamma I$$

has no symmetric (Hermitian) positive definite solution X.
If $(1+\frac{1}{\Gamma})^2 \leqslant \frac{4}{\Gamma}\Lambda$, then any symmetric (Hermitian) positive definite X satisfies the Riccati inequality

$$S_c \leqslant \Gamma I.$$

Proof:
From Lemma 6.67 and Lemma 6.72 we already know, that $X_\pm(\alpha)$ are the extremal solutions of the Riccati equation $S_c(X) = \frac{1}{\alpha}I$. We still have to show that both solutions are positive definite and that inequalities (6.78), (6.79) hold. By Lemma 6.67 it is clear, that at least $X_+(\alpha) > O$.

$$\begin{array}{rrcl}
 & X_-(\alpha) & > & O \\
\Longleftrightarrow & P^*T_1(\alpha)P & > & (P^*T_2(\alpha)P)^{1/2} \\
\Longleftrightarrow & T_1(\alpha)CT_1(\alpha) & > & T_2(\alpha) \\
\Longleftrightarrow & \left(\frac{\alpha-1}{2}C^{-1} - D\right) C \left(\frac{\alpha-1}{2}C^{-1} - D\right) & > & T_2(\alpha) \\
\Longleftrightarrow & T_2(\alpha) + D + DCD & > & T_2(\alpha) \\
\Longleftrightarrow & D + DCD & > & O.
\end{array}$$

Next we will show (6.78). Since $X_\pm(\alpha)$ can be uniformly transformed to diagonal form by any orthogonal matrix which consists of an eigenvector basis of W, inequalities (6.78) are essential diagonal problems, i.e., scalar problems. For this purpose consider $w \in [0,1)$, define $x_\pm(\alpha) = (\alpha-1) - (\alpha+1)w \pm \sqrt{(1-w)((\alpha-1)^2 - (\alpha+1)^2 w)}$. It suffices to show that the derivatives satisfy $x'_+(\alpha) > 0, x'_-(\alpha) < 0$. It is easy to verify that

$$x'_\pm(\alpha) = \pm\sqrt{\frac{1-w}{(\alpha-1)^2 - (\alpha+1)^2 w}}\; x_\pm(\alpha).$$

But we already know that $x_\pm(\alpha)$ is always positive, since $X_\pm(\alpha)$ is positive definite.
Next we show (6.79). If $\alpha_1 < \alpha_2$, then $X_-(\alpha_1) < X_+(\alpha_1) < X_+(\alpha_2)$. if $\alpha_1 > \alpha_2$, then $X_-(\alpha_1) < X_-(\alpha_2) < X_+(\alpha_2)$.
The remaining assertions follow immediately from Lemma 6.72 and inequalities (6.78),(6.79). □

Example 6.83 *Consider A_{12} from (6.56) for $p = 3$. For the matrix W we obtain $\lambda \approx$* $1.0036, \Lambda \approx 4.878 \cdot 10^2$. *By Theorem 6.76 we will always have*

$$S_c \geqslant \gamma I \Longrightarrow \gamma \leqslant 5.12 \cdot 10^{-4} I.$$

This bound is prescribed by the problem and independent on the choice of X . By Theorem 6.76 this lower bound cannot be improved whatever the choice of X will be. Even the exact *solution of the Riccati equation must satisfy this bound. Thus this bound is sharp. For the other inequality we will always have*

$$S_c \leqslant 0.681 I.$$

This bound can be improved. 0.681 *will be an upper bound for any choice of X. By Theorem 6.76 we cannot do worse. For the exact solution this upper bound can be moved to the lower bound* $5.12 \cdot 10^{-4}$. *If we choose $X = I$, then we will have*

$$4.05 \cdot 10^{-4} I \leqslant S_c \leqslant 0.678 I.$$

As expected the lower bound must be less than $5.12 \cdot 10^{-4}$. *Likewise the upper bound has to be less than* 0.681, *but unfortunately it is pretty close to it for this choice of X. This yields a large condition number of* 1674. *This will be unsatisfactory for the solution of systems with S_c.*

Theorem 6.76 requires the computation of the square root $(P^* T_2(\alpha) P)^{1/2}$ from (6.73). Since this is very expensive one would like to omit this square root. By Theorem 6.51 we already know that $X = T_1(\alpha)$ almost solves the problem. But $T_1(\alpha)$ still has to be replaced by a block diagonal matrix. Typically a block diagonal approximation X of $T_1(\alpha)$ will be described in quadratic forms by an inequality of the form

$$\delta X \leqslant T_1(\alpha) \leqslant \Delta X.$$

Note that the almost best block diagonal approximation of $T_1(\alpha)$ with respect to the ratio Δ/δ is the block diagonal part of $T_1(\alpha)$ itself (see e.g. [21]).
More sensibly adapted to the situation here is the inequality

$$T_1(\alpha_\Delta) \leqslant X \leqslant T_1(\alpha_\delta), \tag{6.84}$$

for suitable $\alpha_\delta, \alpha_\Delta$. The following Corollary shows how this affects the properties of S_c.

Corollary 6.85 *Using the notation of Theorem 6.76 we assume that $\alpha_\delta \geqslant \alpha_\Delta$ are positive constants such that (6.84) is fulfilled. Then S_c satisfies*

$$S_c \leqslant \frac{2}{\alpha_\Delta + 1} I. \tag{6.86}$$

If in addition $\frac{\alpha_\delta - 1}{\alpha_\delta + 1} \frac{\alpha_\Delta - 1}{\alpha_\Delta + 1} \geqslant 1 - \frac{1}{\Lambda}$, *then*

$$\frac{2}{\alpha_\delta + \alpha_\Delta} I \leqslant S_c. \tag{6.87}$$

Proof:
First we will show (6.86). $X - T_1(\frac{\alpha_\Delta + 1}{2}) \geqslant \frac{\alpha_\Delta - 1}{4} C^{-1} \geqslant O$. From this it follows that

$$N(S_c^{-1} - \frac{\alpha_\Delta + 1}{2} I) N^* \geqslant \left(\frac{\alpha_\Delta - 1}{4}\right)^2 C^{-1} - T_2(\frac{\alpha_\Delta + 1}{2}) = \frac{\alpha_\Delta + 1}{2} D \geqslant O.$$

Next we will show (6.87). We have $\frac{\alpha_\delta-\alpha_\Delta}{4}C^{-1} \geqslant X - T_1(\frac{\alpha_\delta+\alpha_\Delta}{4}) \geqslant \frac{\alpha_\Delta-\alpha_\delta}{4}C^{-1}$. From this it follows that

$$\begin{aligned}
N(S_c^{-1} - \frac{\alpha_\delta + \alpha_\Delta}{2}I)N^* &\leqslant \left(\frac{\alpha_\delta - \alpha_\Delta}{4}\right)^2 C^{-1} - T_2(\frac{\alpha_\delta + \alpha_\Delta}{2}) \\
&\leqslant -\frac{(\alpha_\delta - 1)(\alpha_\Delta - 1)}{4}C^{-1} + \frac{\alpha_\delta + \alpha_\Delta}{2}D \\
&= -\frac{(\alpha_\delta - 1)(\alpha_\Delta - 1)}{4}\bar{D}^{-1} + \frac{(\alpha_\delta + 1)(\alpha_\Delta + 1)}{4}D \\
&\leqslant 0.
\end{aligned}$$

□

Remark: For the special case $\alpha_\delta = \alpha_\Delta \equiv \frac{1}{\gamma}$ we obtain $\gamma I \leqslant S_c \leqslant \frac{2\gamma}{1+\gamma}I$, where $0 < \gamma < 1$ and typically $\gamma \ll 1$. So omitting the square root part of the solution is not critical, $T_1(\frac{1}{\gamma})$ is almost optimal with respect to these inequalities.
By Corollary 6.85 it is advisable to scale X such that $\alpha_\delta \approx \frac{1}{\gamma}$ or a little bit less than 1. We cannot improve the condition number of S_c by this scaling, but we can avoid that the eigenvalues are too much scaled in the direction of 0.

Example 6.88 *Consider A_{12} from (6.56) for $p = 3$. For $X = T_1(\alpha)$ we will get*

$$5.12 \cdot 10^{-4} I \leqslant S_c \leqslant 1.02 \cdot 10^{-3} I.$$

The upper bound is less than twice as much as the lower bound. Thus $T_1(\alpha)$ is almost optimal. If we choose X as the block diagonal part of $T_1(\alpha)$, then we get

$$5.5 \cdot 10^{-5} I \leqslant S_c \leqslant 1.94 \cdot 10^{-3} I.$$

The resulting condition number will be 35.3. This is many times better than for the choice $X = I$, where we had a condition number of 1674. If we scale X such that $\alpha_\delta \approx 1/4$, then we obtain

$$3.16 \cdot 10^{-4} I \leqslant S_c \leqslant 1.29 \cdot 10^{-2} I.$$

This is slightly worse, but both bounds are approximately 10 times larger.

In this section we have discussed the expression $R(X)$ from (6.12) for the positive definite case and derived approximate solutions in the sense of quadratic forms. Moreover we have shown the sharpness of the results and how approximate solutions affect the resulting coupling system. It remains to summarize the results in an algorithm.

6.7 Algorithm

To apply the theory presented in Section 1–6 we will present an abstract algorithm. The algorithm essentially consists of computing $T_1(\alpha)$. By Theorem 6.51, $T_1(\alpha)$ is an almost

optimal solution for the Riccati–equation (6.39) for nonsingular C from Lemma 6.19 and in Section 3 we have given sufficient conditions to achieve the nonsingularity for C. For simplicity we restrict ourselves to classes of matrices which fulfil the assumptions of Corollary 6.35.
One way to approximate $T_1(\alpha)$ by X could be to choose X as block diagonal part of $T_1(\alpha)$. To choose the block diagonal part of $T_1(\alpha)$ will most likely not be the best choice in the unsymmetric case.

Algorithm 6.89 (Compute Approximate Block Diagonal Solution)
For any $\{q,r\} \in \mathfrak{I}$ with $q \leqslant s < r$ assume that A_{qr}, A_{rq} are factorized with respect to (6.28) and (6.29).
Perform an LU decomposition of $A_{11}, \dots, A_{pp}$.
Define for $q = 1, \dots, p$, F_q, G_q by

$$F_q = \left[\cdots F_q^{\{q,r\}} \cdots\right]_{r:\,\{q,r\}\in\mathfrak{I}}, \; G_q = \begin{bmatrix} \vdots \\ G_q^{\{q,r\}} \\ \vdots \end{bmatrix}_{r:\,\{q,r\}\in\mathfrak{I}} .$$

Using the LU–decomposition of A_{ii} compute $\bar{D}, D$ by

$$\bar{D} = \operatorname{diag}(G_1 A_{11}^{-1} F_1, \dots, G_s A_{ss}^{-1} F_s), \; D = \operatorname{diag}(G_{s+1} A_{s+1,s+1}^{-1} F_{s+1}, \dots, G_p A_{pp}^{-1} F_p).$$

Compute an estimate Λ for $1/\sigma_{\min}(I - \bar{D}D)$ and choose α such that

$$(\alpha+1)^2 \geqslant 4\alpha\Lambda.$$

Define the block diagonal matrix $X = \operatorname{diag}\left(X^{\mathbf{i},\mathbf{i}}\right)_{\mathbf{i}\in\mathfrak{I}}$ by $X^{\mathbf{i},\mathbf{i}} = T_1(\alpha)^{\mathbf{i},\mathbf{i}}$.
Perform an LU decomposition $N^{\mathbf{i},\mathbf{i}} \cdot \bar{N}^{\mathbf{i},\mathbf{i}} = X^{\mathbf{i},\mathbf{i}}$ of $X^{\mathbf{i},\mathbf{i}}$.

Note that in the symmetric positive definite case we can replace the LU decomposition by the Cholesky decomposition.

The bottle neck in this algorithm is the computation of Λ which may become relatively expensive. There are several ways to compute an estimate, see e.g. [41],p.351ff. The main problem will be that one would not like to solve systems with $I - \bar{D}D$, since this will be as expensive as the solution with of systems with the coupling systems S_c. An additional problem is, that $\bar{D}$ and D are distributed over the processors when the method is implemented on a parallel computer. From this point of view the inverse iteration[41],p.383ff is not suitable. Essentially we can only perform matrix–vector multiplications with $\bar{D}$ and D. To get an estimate for α one can use Lanczos' method [67]. The problem in using Lanczos' method will be that the convergence of the approximate eigenvalues obtained from the Lanczos method may be slow. In this case the estimate for Λ can be too small. The situation could be more improved using implicitly restarted Lanczos methods [81],[19],[56]. It has been observed that for implicitly restarted Lanczos methods the convergence of the extremal eigenvalues will be typically faster than for the normal Lanczos process.

In addition we will discuss a heuristic variant of Algorithm 6.89, which will not need α. A simple observation can help to have an estimate for α. Under the assumptions of Lemma 6.48 we have

$$T_1(\alpha) = \frac{\alpha+1}{2} C^{-1} (I - E_1), \; \|E_1\| \leqslant \delta < 1.$$

From this it follows that

$$\|T_1(\alpha)^{-1}\| \leqslant \frac{2}{(\alpha+1)(1-\delta)} \|(I - \bar{D}D)^{-1}\| \, \|\bar{D}\| \leqslant \frac{\alpha+1}{2\alpha} \frac{\delta}{1-\delta} \|\bar{D}\|.$$

If in (6.49) α is chosen such that

$$\frac{(\alpha+1)^2}{4\alpha} = \|(I - \bar{D}D)^{-1}\|/\delta,$$

then we also have a lower bound for $\|T_1(\alpha)^{-1}\|$.

$$\|(I-\bar{D}D)^{-1}\| = \|C\bar{D}^{-1}\| = \frac{\alpha+1}{2}\|T_1(\alpha)^{-1}(I-E_1)\bar{D}^{-1}\| \leqslant \frac{(\alpha+1)(1-\delta)}{2} \|T_1(\alpha)^{-1}\| \, \|\bar{D}^{-1}\|.$$

$$\Longrightarrow \|T_1(\alpha)^{-1}\| \geqslant \frac{\alpha+1}{2\alpha} \frac{\delta}{1-\delta} \frac{1}{\|\bar{D}^{-1}\|}.$$

For large α and sufficient small δ we can assume that $\frac{\alpha+1}{2\alpha}\frac{\delta}{1-\delta} \leqslant 1$ or at least $\frac{\alpha+1}{2\alpha}\frac{\delta}{1-\delta} \approx 1$. Typically $\|\bar{D}\| \geqslant 1$ and $\|\bar{D}^{-1}\| \gg 1$. Unless $\|\bar{D}\| \leqslant 1$, any sufficient large α which ensures that

$$\|T_1(\alpha)^{-1}\| \leqslant 1$$

should be satisfactory.
If α is large, then by (6.50) we have $T_1(\alpha) \approx \frac{\alpha+1}{2} C^{-1} = \frac{\alpha+1}{2}(\bar{D}^{-1} - D)$. To have an approximate block diagonal solution X we suggest to set

$$X = \beta \, \text{blockdiag}(\bar{D}^{-1} - D),$$

where β is chosen such that $\|X^{-1}\| \leqslant \min\{1, \|\bar{D}\|\}$. This choice of X will be compared in the numerical examples with X from Algorithm 6.89.

Algorithm 6.90 (Heuristic Block Diagonal Solution)
For any $\{q,r\} \in \Im$ with $q \leqslant s < r$ assume that A_{qr}, A_{rq} are factorized with respect to (6.28) and (6.29).
Perform an LU decomposition of $A_{11}, \ldots, A_{pp}$.
Define for $q = 1, \ldots, p$, F_q, G_q by

$$F_q = \left[\cdots F_q^{\{q,r\}} \cdots\right]_{r:\,\{q,r\}\in\Im}, \quad G_q = \begin{bmatrix} \vdots \\ G_q^{\{q,r\}} \\ \vdots \end{bmatrix}_{r:\,\{q,r\}\in\Im}.$$

Using the LU–decomposition of A_{ii} compute $\bar{D}, D$ by

$$\bar{D} = \operatorname{diag}(G_1 A_{11}^{-1} F_1, \ldots, G_s A_{ss}^{-1} F_s), \ D = \operatorname{diag}(G_{s+1} A_{s+1,s+1}^{-1} F_{s+1}, \ldots, G_p A_{pp}^{-1} F_p).$$

Define the block diagonal matrix $X = \operatorname{diag}\left(X^{\mathbf{i},\mathbf{i}}\right)_{\mathbf{i}\in\Im}$ by $X^{\mathbf{i},\mathbf{i}} = \left(\bar{D}^{-1} - D\right)^{\mathbf{i},\mathbf{i}}$.
Compute an estimate β for $1/\|X^{-1}\|$ and set $X = \beta X$.
Perform an LU decomposition $N^{\mathbf{i},\mathbf{i}} \cdot \bar{N}^{\mathbf{i},\mathbf{i}} = X^{\mathbf{i},\mathbf{i}}$ of $X^{\mathbf{i},\mathbf{i}}$.

Remark: Common packages for LU decompositions like LINPACK or LAPACK provide estimates for the norm of the inverse when performing the LU decomposition with an additional overhead, which only is in the magnitude of solving one system with the factors from the LU decomposition.
For the positive definite case we can replace the LU decomposition everywhere by the Cholesky decomposition, since $C^{-1} = \bar{D}^{-1} - D$ is positive definite.

To take the block diagonal part of C^{-1} or the block diagonal part of $T_1(\alpha)$ in Algorithm 6.89,6.90 will most likely not be the best choice in the unsymmetric case. Other choices may be better. In order to improve this choice we will introduce an additional scaling. Since $N \cdot \bar{N}$ should approximate $T_1(\alpha)$ ($\frac{\alpha+1}{2}C^{-1}$ respectively) we should ensure to have $N^{-1}T_1(\alpha)\bar{N}^{-1}$ close to the identity. We can try decrease the norm of the off–diagonal part of this matrix by an additional balancing. In theory, by Theorem 6.51 it suffices to reduce $\frac{2}{\alpha+1}CX - I$ in a norm. In practice we can at most reduce $\frac{\alpha+1}{2}X^{-1}C^{-1} - I$ in some norm. Now any decomposition $N\bar{N} = X$ can be changed to $(NE)(E^{-1}\bar{N})$ without changing X. What we can do is to construct E in order to reduce $\|E^{-1}\frac{\alpha+1}{2}N^{-1}C^{-1}\bar{N}^{-1}E - I\|$ or $\|E^{-1}N^{-1}T_1(\alpha)\bar{N}^{-1}E - I\|$. To reduce the norm by similarity transformation see [66],[68]. The main idea behind balancing is to successively transform a matrix by equivalence transformation such that locally the row sum and column sum of the absolute values of a given matrix will be the same. This is done successively starting from the first row/column to the last row column.

Summary

In this chapter we have discussed modifications for block Jacobi splittings to improve the properties of the coupling system. It has turned out that under the relatively general condition $\bar{M}S_J^{-1}L = O, \bar{L}S_J^{-1}M = O$ in Lemma 6.36 and an additional nonsingularity condition

on S_J, we can rewrite the modified coupling system as a linear quadratic expression of the form $(X+D)+(X+D)C(X+D)$. This has lead to finding solutions of algebraic Riccati equations. The condition $\bar{M}S_J^{-1}L = O, \bar{L}S_J^{-1}M = O$ is fulfilled at least by the class of block 2–cyclic matrices. The information about the relabelling of the diagonal blocks to obtain $\bar{M}S_J^{-1}L = O, \bar{L}S_J^{-1}M = O$ can be read as part of a preprocessing step.
For the case that C is nonsingular it has been shown that solutions of the Riccati equation exist.
An exact solution requires the computation of a matrix square root. We have derived an approximate solution for the Riccati equation which is close to the exact solution without having to use of the matrix square root. These approximate solutions still have to be replaced by block diagonal matrices.
For the symmetric positive definite case we have shown sharper results. They show that the smallest eigenvalue of S_c has a lower bound which cannot be improved even for the exact solution. The matrix $T_1(\alpha)$ is almost optimal with respect to the condition number of S_c. Moreover to choose X as the block diagonal part of $T_1(\alpha)$ is almost the best approximation among all block diagonal matrices.

The following problems are still open.
First it is still an open problem how the theory can be extended if the condition $\bar{M}S_J^{-1}L = O, \bar{L}S_J^{-1}M = O$ is not fulfilled.
More critical is the question how this theory can be generalized to the case when S_J is singular. In principle we can express S_J in terms of $S = S_J + LX_0\bar{L} + MX_0^{-1}\bar{M}$ for a given choice of X_0. The initial choice of X_0 should ensure that at least S is nonsingular. S is the Schur–complement of the matrix

$$\left(\begin{array}{c|cc} S_J & -L & -M \\ \hline \bar{L} & X_0^{-1} & O \\ \bar{M} & O & X_0 \end{array}\right)$$

and the matrices $[S_J, -L, -M]$ as well as $\left[S_J^T, \bar{L}^T, \bar{M}^T\right]$ have full rank, since A is nonsingular and A satisfies

$$A = [S_J, -L, -M] \begin{bmatrix} I \\ \bar{L} \\ \bar{M} \end{bmatrix} = [I, -L, -M] \begin{bmatrix} S_J \\ \bar{L} \\ \bar{M} \end{bmatrix}.$$

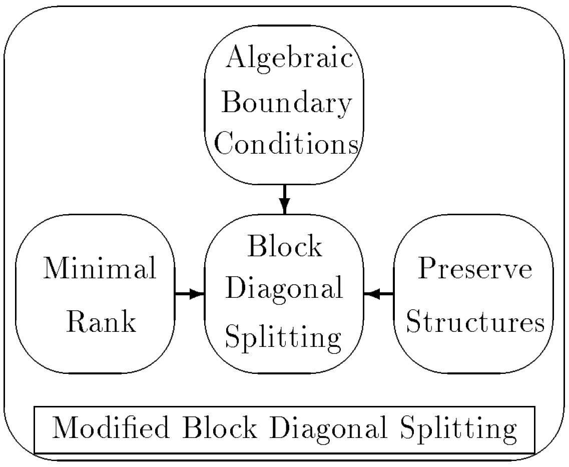
In [32],[33] it has been shown that under these conditions there exist matrices $X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}$ which minimize

$$\| \left(\begin{array}{c|cc} S_J & -L & -M \\ \hline \bar{L} & X_{11} & X_{12} \\ \bar{M} & X_{21} & X_{22} \end{array}\right) \|_2 \text{ or } \| \left(\begin{array}{c|cc} S_J & -L & -M \\ \hline \bar{L} & X_{11} & X_{12} \\ \bar{M} & X_{21} & X_{22} \end{array}\right)^{-1} \|_2.$$

This shows that computing a well–conditioned $S = S_J + LX_0\bar{L} + MX_0^{-1}\bar{M}$ is closely related to the completion problem [32],[33]. In this case the theory has to be adapted.
Another problem in general is the nonsingularity requirement for $C = \bar{L}A^{-1}L,\ \bar{D} =$

$\bar{L}S_J^{-1}L$. For this it is only necessary but not sufficient to have full rank matrices $L, \bar{L}$. For some classes of matrices we have shown that this is already sufficient, but in general it will be necessary to compute the left and right null space of $\bar{D}$.
It is also open, if the nice relations between the left/right null space of C and $\bar{D}$ in Corollary 6.24 have an analogy if S_J is singular.
Finally it is open how we have to replace the approximate solution $T_1(\alpha)$ by an appropriate block diagonal matrix X in Algorithm 6.89. To choose the block diagonal part of $T_1(\alpha)$ will most likely not be the best choice in the unsymmetric case.

We have introduced modified block diagonal splittings and discussed modifications which can be read as some kind of algebraic boundary conditions. The manipulations discussed in Chapter 5 and Chapter 6 on block diagonal Splittings can be summarized in the following table.


Algebraic Boundary Conditions
Minimal Rank
Block Diagonal Splitting
Preserve Structures
Modified Block Diagonal Splitting

From our list of questions on page 10 we now have to establish a parallel model in order to give an answer to question 4 on page 10.

Chapter 7

Parallel Treatment of the Coupling System

In this chapter we will discuss the treatment of the Sherman–Morrison–Woodbury formula (1.4) on a parallel architecture. Here we will only consider the case of block diagonal splittings that have been discussed in Chapter 5 and 6. To do this let us assume that we have p processors each of them having its own memory. The communication should be done by message passing.

A distribution of a matrix A over the processors will be defined. We will briefly comment on how this model of distribution can be affected by preprocessing, that is a reordering of the initial system before its distribution is fixed.

According to the distribution of A we will examine the representation of the coupling system S_c from (1.4) and its related distribution over the processors.

To have a better understanding about the coupling system S_c we will examine the corresponding block graph and give two ways to derive the block graph of S_c from the block graph of A.

The distribution of S_c and its representation involve a natural overlapping distribution of vectors which are related to the coupling system. The concept of adding type vectors and overlapping type vectors which is well–known in finite element methods for partial differential equations [55],[45],[46],[6] can be transferred to the situation here.

We will comment on the consequences of the representation of S_c for solving systems with S_c.

7.1 Distribution of the System and Preprocessing

For the whole chapter we will assume the following distribution of a given matrix A. Let $A \in \mathrm{GL}(n, \mathbb{F})$, p some positive number counting the number of processors. Analogous to

Chapter 5 we will assume that A is partitioned as

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1p} \\ \vdots & & \vdots \\ A_{p1} & \cdots & A_{pp} \end{pmatrix} \tag{7.1}$$

with square diagonal blocks of size $n_1, \ldots, n_p$. We will assume that processor q, $q = 1, \ldots, p$ can directly access the blocks $A_{q,r}, A_{r,q}$, for all $r = 1, \ldots, p$. From this it follows that for any $q \neq r$ $A_{q,r}$ is available for precisely two processors.

Example 7.2 *Let $p = 4$ and*

$$A = \begin{pmatrix} A_{11} & A_{12} & O & O \\ A_{21} & A_{22} & A_{23} & O \\ O & A_{32} & A_{33} & A_{34} \\ O & O & A_{43} & A_{34} \end{pmatrix}.$$

We get the following (overlapping) distribution of A with respect to the processors.

1	2	3	4
A_{11} A_{12}	A_{12}	A_{23}	A_{34}
A_{21}	A_{21} A_{22} A_{23}	A_{32} A_{33} A_{34}	A_{43} A_{34}
	A_{32}	A_{43}	

For sparse matrices the overhead of storing $A_{qr}, q \neq r$ simultaneously on two different processors seems to be acceptable. One could try to reduce the number of off-diagonal blocks $A_{q,r}$ by reordering the rows and columns of A in the form $P^T AP$ by a permutation matrix P. This problem can be described in graph theoretical terms by the undirected graph defined as follows.

Definition 7.3 *Let $A \in \mathrm{GL}\,(n, \mathbb{F})$ as in (7.1).*
Then the undirected scalar graph $G_s(A) = (\mathcal{V}_s(A), \mathcal{E}_s(A)) \equiv (\mathcal{V}_s, \mathcal{E}_s)$ of A is defined by $\mathcal{V}_s = \{1, \ldots, n\}, \mathcal{E}_s = \{\{i, j\} : i \neq j, a_{ij} \neq 0\}$.
The undirected block graph $G_b(A) = (\mathcal{V}_b(A), \mathcal{E}_b(A)) \equiv (\mathcal{V}_b, \mathcal{E}_b)$ of A is defined by $\mathcal{V}_b = \{1, \ldots, p\}, \mathcal{E}_b = \{\{q, r\} : q \neq r, A_{qr} \neq O\}$.

The elements of $\mathcal{V}$ are called vertices and the elements of $\mathcal{E}$ are called edges.

For strategies of permuting A by renumbering the nodes of $\mathcal{V}_s$ we refer to [52], [17], [34], [70], [80]. This preprocessing part is assumed to be done a priori and we will not discuss this in detail here. Denote by $\mathcal{V}_1, \ldots, \mathcal{V}_p$ the sets corresponding to the diagonal blocks $A_{11}, \ldots, A_{pp}$ of the permuted matrix $P^T AP$. At least the following aspects should be handled by the preprocessing step.

1. The number of edges $\mathcal{E}_s \cap (\mathcal{V}_q \times \mathcal{V}_r)$, $q \neq r$, $q, r = 1, \ldots, p$ should be as small as possible.

2. $\#\mathcal{E}_b$ should be as small as possible.

3. G_b should be constructed with respect to the underlying processor topology, that is the physical communication network between the processors.

Of course, not all of these criteria can be satisfied simultaneously. The first criterion is suitable to decrease the rank of the remaining part and therefore the size of the coupling system. The second criterion will have influence on the block graph of the coupling system (see Section 3 for details) and its fill–in. Many nonempty sets $\mathcal{E}_s \cap (\mathcal{V}_q \times \mathcal{V}_r)$ of small size may reduce the size of the coupling system but will require additional communication. In principle communication has to be performed between all processors p, q such that $\{q, r\} \in G_b$, which can be seen in Section 4. So a fewer number of elements in nonempty sets $\mathcal{E}_s \cap (\mathcal{V}_q \times \mathcal{V}_r) \neq \emptyset$ of larger size might be more useful than several sets $\mathcal{E}_s \cap (\mathcal{V}_q \times \mathcal{V}_r) \neq \emptyset$ of small size. Finally the third criterion is necessary to adapt the algebraic problem to the underlying processor system.
The distribution of A over the processors will induce a special overlapping distribution for the coupling system S_c. This will be investigated in the next section.

7.2 Representation of S_c

Based on the distribution of A introduced in Section 1 we will examine the representation of the related coupling system S_c from (1.4) and the corresponding distribution over the processors.
Analogous to Chapter 5 we denote by

$$\mathfrak{I} := \{\{q, r\} : \ q \neq r, \ A_{q,r} \neq O \ \textit{or} \ A_{r,q} \neq O\} \tag{7.4}$$

and assume that the indices of $\mathfrak{I}$ are taken in some fixed order $\mathbf{i}_1, \ldots, \mathbf{i}_s$.

For any $\mathbf{i} = \{q, r\} \in \mathfrak{I}$ we assume that A_{qr}, A_{rq} are factored as

$$-A_{qr} = F^{\mathbf{i}}_q G^{\mathbf{i}}_r, -A_{rq} = F^{\mathbf{i}}_r G^{\mathbf{i}}_q, \tag{7.5}$$

with matrices $F^{\mathbf{i}}_q, (G^{\mathbf{i}}_q)^T \in \mathrm{M}\,(n_q \times n^{\mathbf{i}}, \mathbb{F}), G^{\mathbf{i}}_r, (F^{\mathbf{i}}_r)^T \in \mathrm{M}\,(n^{\mathbf{i}} \times n_r, \mathbb{F})$ for some positive number $n^{\mathbf{i}}$. We define

$$n_c = \sum_{\mathbf{i} \in \mathfrak{I}} n^{\mathbf{i}}. \tag{7.6}$$

Choices of this factorization have already been discussed in Chapter 5. For simplicity we assume that this factorization has been performed in parallel a priori on processor r, q simultaneously.

The ordering $\mathbf{i}_1, \ldots, \mathbf{i}_s$ of the elements in $\mathfrak{I}$ and the corresponding sizes $n^{\mathbf{i}_1}, \ldots, n^{\mathbf{i}_s}$ induce a block partitioning for vectors in $\mathbb{F}^{n_c}$ and matrices in $\mathrm{M}\,(n \times n_c, \mathbb{F})$, $\mathrm{M}\,(n_c \times n, \mathbb{F})$. Partition the identity matrix I_{n_c} of size $n_c \times n_c$ columnwise as

$$I_{n_c} = \left(E^{\mathbf{i}_1}, \ldots, E^{\mathbf{i}_s}\right), \tag{7.7}$$

where $E^{\mathbf{i}_l}$ denotes $n^{\mathbf{i}_l}$ unit vectors one after another. This notation has already been introduced in (5.5)–(5.10).

We define F, G analogous to (5.13) by

$$(7.8)\qquad F = \sum_{\mathbf{i}=\{q,r\}\in\mathfrak{I}} (E_q F_q^{\mathbf{i}} + E_r F_r^{\mathbf{i}})(E^{\mathbf{i}})^T,\ G = \sum_{\mathbf{i}=\{q,r\}\in\mathfrak{I}} E^{\mathbf{i}}(G_q^{\mathbf{i}} E_q^T + G_r^{\mathbf{i}} E_r^T)$$

and the block diagonal matrix $S = \text{diag}\,(S_{11}, \ldots, S_{pp})$ by

$$(7.9)\qquad S_{qq} = A_{qq} + \sum_{\mathbf{i}=\{q,r\}\in\mathfrak{I}} F_q^{\mathbf{i}} G_q^{\mathbf{i}},\ q = 1, \ldots, p.$$

This definition corresponds to the definition of S in (5.15). Here we will assume that S is invertible.

S_{qq} can be computed in parallel since $F_q^{\mathbf{i}}, G_q^{\mathbf{i}}$ are available on processor q.

From this definition we obtain a splitting

$$(7.10)\qquad A = S - FG.$$

For an example of this splitting we refer to Example 5.16.

The corresponding coupling system for the Sherman–Morrison–Woodbury formula (1.4) will be

$$(7.11)\qquad S_c = I - GS^{-1}F.$$

The definition of F, G induces a block partitioning for $S_c = (S_c^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{I}}$ with blocks $S_c^{\mathbf{i},\mathbf{j}}$ of size $n^{\mathbf{i}} \times n^{\mathbf{j}}$. Analogous to Chapter 5 we will superpose indices for matrices and vectors which correspond in their size and partitioning to S_c. These indices are always elements of $\mathfrak{I}$.

The distribution of the blocks of A induces in a natural way a distribution for the coupling system S_c.

Lemma 7.12 *Let $S_c = (S_c^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{I}} = I - GS^{-1}F \in \mathrm{GL}\,(n_c, \mathbb{F})$. With respect to the partitioning from (7.7), $I - S_c$ can be written as sum of p elementary matrices $M_1, \ldots, M_p$,*

$$(7.13)\qquad S_c = I - \sum_{q=1}^{p} M_q,$$

where each $M_q = (M_q^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{I}}$ has the following properties:

$$(7.14)\qquad M_q = \sum_{\mathbf{i},\mathbf{j}\in\mathfrak{I}:\ q\in\mathbf{i}\cap\mathbf{j}} E^{\mathbf{i}}(G_q^{\mathbf{i}} S_{qq}^{-1} F_q^{\mathbf{j}})(E^{\mathbf{j}})^T.$$

M_q is available on processor q without communication.
$M_q^{\mathbf{i},\mathbf{j}} = \mathrm{O}$, if $q \notin \mathbf{i} \cap \mathbf{j}$.
If $M_q^{\mathbf{i},\mathbf{j}} \neq \mathrm{O}$ and $M_r^{\mathbf{i},\mathbf{j}} \neq \mathrm{O}$ for $q \neq r$, then $\mathbf{i} = \mathbf{j} = \{q, r\}$, i.e., two different elementary matrices M_q, M_r overlap at most in one diagonal block, provided that $\{q, r\} \in \mathfrak{I}$.

Proof:
We can write the coupling system S_c as

$$\begin{aligned}
S_c &= I - GS^{-1}F \\
&= I - \sum_{q=1}^{p} GE_q S_{qq}^{-1} E_q^T F \\
&= I - \sum_{q=1}^{p} \Big(\sum_{\mathbf{i}\equiv\{r,q\}\in\mathfrak{I}} E^{\mathbf{i}} G_q^{\mathbf{i}} \Big) S_{qq}^{-1} \Big(\sum_{\mathbf{j}\equiv\{s,q\}\in\mathfrak{I}} F_q^{\mathbf{j}} (E^{\mathbf{j}})^T \Big) \\
&= I - \sum_{q=1}^{p} \sum_{\mathbf{i}\equiv\{r,q\},\mathbf{j}\equiv\{s,q\}\in\mathfrak{I}} E^{\mathbf{i}} (G_q^{\mathbf{i}} S_{qq}^{-1} F_q^{\mathbf{j}})(E^{\mathbf{j}})^T \\
&= I - \sum_{q=1}^{p} \sum_{\mathbf{i},\mathbf{j}\in\mathfrak{I}:\ q\in\mathbf{i}\cap\mathbf{j}} E^{\mathbf{i}} (G_q^{\mathbf{i}} S_{qq}^{-1} F_q^{\mathbf{j}})(E^{\mathbf{j}})^T \\
&= I - \sum_{q=1}^{p} M_q .
\end{aligned}$$

Thus $I - S_c$ can be written as sum of p elementary matrices $M_q,\ q = 1, \dots, p$.

M_q is obviously computable on processor q without communication, since $S_{qq}, F_q^{\mathbf{i}}, G_q^{\mathbf{j}}$ are stored on processor q.

By the definition of M_q it follows immediately that M_q can have nonzero blocks only in those positions $M_q^{\mathbf{i},\mathbf{j}}$, $\mathbf{i},\mathbf{j} \in \mathfrak{I}$, where $\mathbf{i}$ and $\mathbf{j}$ both contain q.

Consider $q \neq r, 1 \leqslant q, r \leqslant p$. M_q can have non trivial blocks only in positions $M_q^{\mathbf{i},\mathbf{j}}$, if $q \in \mathbf{i} \cap \mathbf{j}$. Analogously M_r can have non trivial blocks only in positions $M_r^{\mathbf{i},\mathbf{j}}$, if $r \in \mathbf{i} \cap \mathbf{j}$. To have a common block $M_q^{\mathbf{i},\mathbf{j}}, M_r^{\mathbf{i},\mathbf{j}}$, we need $\{q, r\} \subset \mathbf{i} \cap \mathbf{j}$. But this is only possible if $\mathbf{i} = \mathbf{j} = \{q, r\} \in \mathfrak{I}$. □

Remark: By Lemma 7.12 we get nonzero blocks of S_c (at most) for all nonempty intersections $\{q, r\} \cap \{s, t\}$, $\{q, r\}, \{s, t\} \in \mathfrak{I}$.

Example 7.15 *Let* $p = 6$ *and let the set* $\mathfrak{I} = \{\{1,2\},\{2,3\},\{4,5\},\{5,6\},\{1,4\},\{2,5\},\{3,6\}\}$ *be given. According to Lemma 7.12,* $I - S_c$ *is a sum of* 6 *matrices* $M_1, \dots, M_6$*, each of them present on the corresponding processor without communication. We use for each* M_q *a different symbol in the following way:*

M_1	M_2	M_3	M_4	M_5	M_6
$+$	$*$	$\square$	$\bigcirc$	$\diamond$	$\star$

Then the distribution of the blocks of S_c can be described by the following pattern:

	1 2	2 3	4 5	5 6	1 4	2 5	3 6
1, 2	$\ast\!\!+$	$\ast$			$+$	$\ast$	
2, 3	$\ast$	⊠				$\ast$	□
4, 5		◈	◇	○	◇		
5, 6		◇	◈		◇		⋆
1, 4	$+$		○		⊕		
2, 5	$\ast$	$\ast$	◇	◇		◈	
3, 6		□		⋆			⊠

The picture illustrates that only the diagonal blocks can overlap. Two different elementary matrices M_q, M_r overlap precisely in one diagonal block, if $\{q, r\} \in \mathfrak{I}$. Otherwise they do not overlap.

Lemma 7.12 and Example 7.15 show that $M_1, \ldots, M_p$ have several blocks which are zero. Moreover, by Lemma 7.12 we can fix those blocks which are possibly nonzero, i.e. we can reduce any M_q to its nontrivial part. This will be done in the next corollary.

Corollary 7.16 *Using the notation of Lemma 7.12 we define for any $q = 1, \ldots, p$ the matrix K_q by*

(7.17) $$K_q = \left(\cdots E^{\{q,r\}} \cdots\right)_{r:\{q,r\}\in\mathfrak{I}},$$

where $E^{\{q,r\}}$ corresponds to the partitioning of the identity matrix in (7.7). Set $\hat{M}_q = K_q^T M_q K_q$.
Then $M_q = K_q \hat{M}_q K_q^T$ and $\hat{M}_q$ equals $\hat{M}_q = (M_q^{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j}\in\mathfrak{I}:\, q\in\mathbf{i}\cap\mathbf{j}}$. In addition we have $\sum_{q=1}^{p} K_q K_q^T = 2I$.

Proof:
From Lemma 7.12 it immediately follows that M_q only has nonzero blocks $M_q^{\mathbf{i},\mathbf{j}}$ for $q \in \mathbf{i} \cap \mathbf{j}$. By definition, the block columns of K_q are the corresponding block columns and so $K_q^T M_q K_q$ is just the reduction of M_q to those blocks which satisfy $q \in \mathbf{i} \cap \mathbf{j}$. For the sum $\sum_{q=1}^{p} K_q K_q^T$ we note that any block unit vector $E^{\mathbf{i}} = E^{\{q,r\}}$ will appear precisely twice. Once as block column of K_q and once as block column of K_r. From this it follows that $\sum_{q=1}^{p} K_q K_q^T = 2\sum_{\mathbf{i}\in\mathfrak{I}} E^{\mathbf{i}}(E^{\mathbf{i}})^T = 2I$. □

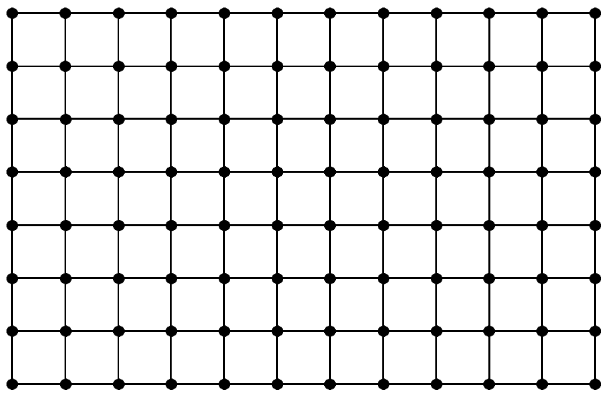
Example 7.18 *We continue Example 7.15. By Corollary 7.16 we obtain $K_1 = \left(E^{\{1,2\}}, E^{\{1,4\}}\right)$, $K_2 = \left(E^{\{1,2\}}, E^{\{2,3\}}, E^{\{2,5\}}\right)$, $K_3 = \left(E^{\{2,3\}}, E^{\{3,6\}}\right)$, $K_4 = \left(E^{\{4,5\}}, E^{\{1,4\}}\right)$, $K_5 = \left(E^{\{4,5\}}, E^{\{5,6\}}, E^{\{2,5\}}\right)$, $K_6 = \left(E^{\{5,6\}}, E^{\{3,6\}}\right)$. The pattern in Example 7.15 has already illustrated that $M_1, \ldots, M_q$ only have blocks in those positions associated with $K_1, \ldots, K_6$.*

In this section we have shown that the distribution of A over the processors has lead to an interesting representation of S_c as sum of p elementary matrices. As Lemma 7.12 has already shown, the block graph of S_c is not arbitrary. In the next section we will derive the block graph of S_c from the block graph of A.

7.3 Deriving the Block Graph of S_c

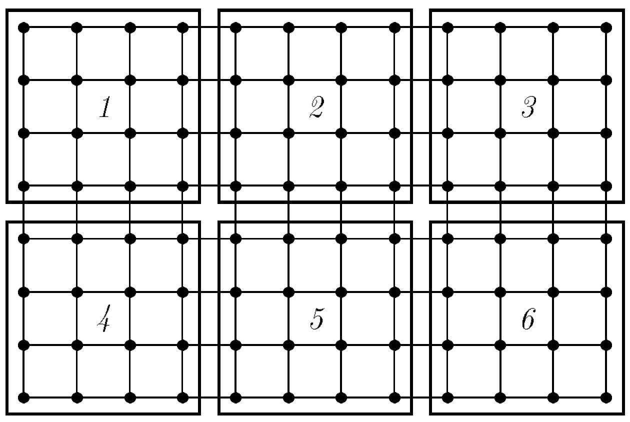
In this section we will derive the block graph of S_c from the block graph of A.

Example 7.19 *Assume that the scalar graph $G_s(A) = (\mathcal{V}_s, \mathcal{E}_s)$ of A can be represented by the following rectangular mesh, where the vertices in $\mathcal{V}_s$ are on the crossings and the edges of $\mathcal{E}_s$ are the lines between the crossings.*

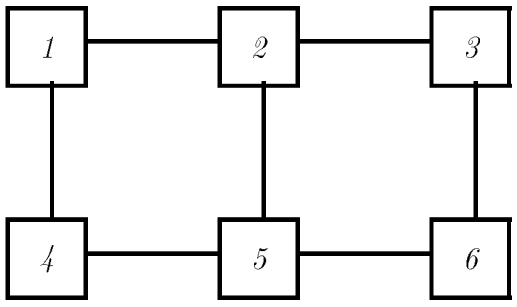


Consider the scalar graph $G_s(A)$ of A. Next we will assume that the set of vertices V_s is written as disjoint union of p nonempty subsets, $\mathcal{V}_s = \mathcal{V}_1 \dot{\cup} \cdots \dot{\cup} \mathcal{V}_p$. These p subsets induce subgraphs or subdomains $(\mathcal{V}_q, \mathcal{E}_{qq})$, where $\mathcal{E}_{qq} = \mathcal{E}_s \cap (\mathcal{V}_q \times \mathcal{V}_q)$. Beside these edge sets $\mathcal{E}_{11}, \ldots, \mathcal{E}_{pp}$ we have for all $q \neq r$ $\mathcal{E}_{qr} = \mathcal{E}_s \cap ((\mathcal{V}_q \times \mathcal{V}_r) \cup (\mathcal{V}_r \times \mathcal{V}_q))$. Then $\mathcal{E}_s$ can be written as disjoint union of all $\mathcal{E}_{qr}$.
Since we have assumed that A has already a given block partitioning the sets $\mathcal{V}_1, \ldots, \mathcal{V}_p$ will be the numbers $\{1, \ldots, n_1\}, \{n_1 + 1, \ldots, n_1 + n_2\}, \ldots, \{\sum_{i=q}^{p-1} n_q + 1, \ldots, \sum_{i=q}^{p} n_q\}$, We set $\mathcal{V}_b = \{1, \ldots, p\}$ and the corresponding edge set $\mathcal{E}_b$ can be obtained from the scalar edge sets by $\mathcal{E}_b = \{\{q, r\} : q \neq r, \mathcal{E}_{qr} \neq \emptyset\}$. This gives us precisely the block graph of A with respect to the given partitioning (7.1).

Example 7.20 *We consider Example 7.19 and assume that the nodes of $\mathcal{V}_1, \ldots, \mathcal{V}_6$ can be characterized by the following 6 overlayed rectangles.*



Now the block graph can be obtained by considering only the rectangles as nodes. An edge in the block graph exists, if there exists a scalar edge between different rectangles.



Using the definition of a block graph of A we can see that

$$\mathfrak{J} = \mathcal{E}_b, \tag{7.21}$$

i.e., the labels for the blocks of S_c are the edges of the block graph of A. Consequently the number of diagonal blocks $\#\mathfrak{J}$ in S_c is precisely the number of edges in the block graph of A.

We can obtain a first abstract description of the block graph of S_c in graph theoretical terms.

Lemma 7.22 *Consider $A \in \mathrm{GL}\,(n,\mathbb{F})$ with the block partitioning from (7.1) and the splitting $A = S - FG$ from (7.10),where S should be nonsingular. Set $\mathcal{P}_b(A)$ the set of all paths in $G_b(A)$ of length 2, i.e.,*

$$\mathcal{P}_b(A) := \{\, \{\{q,r\},\{s,t\}\} : \ \{q,r\},\{s,t\} \in \mathcal{E}_b(A),\ \#(\{q,r\} \cap \{s,t\}) = 1\}. \tag{7.23}$$

Let $S_c = I - GS^{-1}F$ and consider for S_c the induced block partitioning from (7.7).
Then the block graph of S_c can be obtained from the block graph of A in the following way (by using the elements of $\mathcal{E}_b(A)$ as labels instead of $1,\ldots,\#\mathcal{E}_b(A)$):

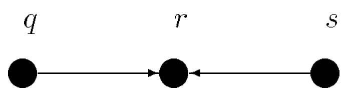
$$\mathcal{V}_b(S_c) = \mathcal{E}_b(A),\ \mathcal{E}_b(S_c) \subseteq \mathcal{P}_b(A). \tag{7.24}$$

Proof:
From Lemma 7.12 we know that $S_c = I - \sum_{q=1}^{p} M_q$. For any matrix M_q, an off-diagonal block $M_q^{\mathbf{i},\mathbf{j}}$ can be different from O, only if $\mathbf{i} \cap \mathbf{j} = \{q\}$.
For $q \neq r$, M_q, M_r do not have any common off–diagonal block. It follows that up to a sign the off-diagonal blocks of S_c are those of $M_1,\ldots,M_p$. Consequently $M_q^{\mathbf{i},\mathbf{j}} \neq$ O or $M_q^{\mathbf{j},\mathbf{i}} \neq$ O for $\mathbf{i} \neq \mathbf{j}$ can at most occur if $\mathbf{i} \cap \mathbf{j} = \{q\}$. So the off-diagonal blocks of S_c are essentially characterized by

$$\bigcup_{q=1}^{p} \{\{\mathbf{i},\mathbf{j}\} : \ \mathbf{i},\mathbf{j} \in \mathfrak{J}, \mathbf{i} \cap \mathbf{j} = \{q\}\} = \bigcup_{q=1}^{p} \{\{\mathbf{i},\mathbf{j}\} : \ \mathbf{i},\mathbf{j} \in \mathcal{E}_b(A), \mathbf{i} \cap \mathbf{j} = \{q\}\}.$$

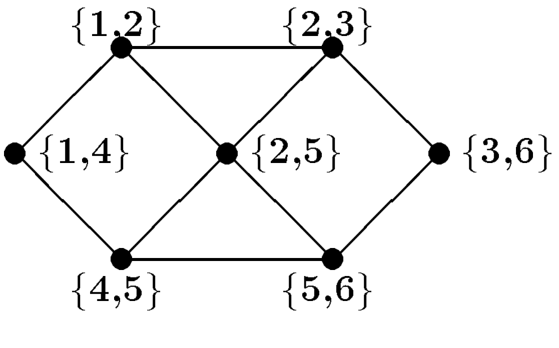
This is obviously $\mathcal{P}_b(A)$. □

Remark: Typically we have $\mathcal{E}_b(S_c) = \mathcal{P}_b(A)$. The reason that in (7.24) only $\mathcal{E}_b(S_c) \subseteq \mathcal{P}_b(A)$ is asserted, is that there are a few exceptions where $\mathcal{E}_b(S_c) \neq \mathcal{P}_b(A)$ can occur. Consider $\mathbf{i} = \{q,r\}, \mathbf{j} = \{r,s\} \in \mathfrak{I}$ with $q < r < s$ and the block entries $S_c^{\mathbf{i},\mathbf{j}} = -G_r^{\{q,r\}} S_{rr}^{-1} F_r^{\{r,s\}}$, $S_c^{\mathbf{j},\mathbf{i}} = -G_r^{\{r,s\}} S_{rr}^{-1} F_r^{\{q,r\}}$ of S_c. If $A_{qr} \neq O$ and $A_{rs} \neq O$, then we have $G_r^{\{q,r\}} \neq O$ and $F_r^{\{r,s\}} \neq O$. So we can expect that $S_c^{\mathbf{i},\mathbf{j}} \neq O$. Obviously one can construct counterexamples where $G_r^{\{q,r\}} \neq O$, $S_{rr}^{-1} \neq O$, $F_r^{\{r,s\}} \neq O$ but $G_r^{\{q,r\}} S_{rr}^{-1} F_r^{\{r,s\}} = O$. Analogously one can proceed for $S_c^{\mathbf{j},\mathbf{i}}$. Another exception is, when $A_{qr} \neq O$, $A_{rs} = O$, $A_{rq} = O$ and $A_{sr} \neq O$. In this case the undirected block graph of A contains the edges $\{q,r\},\{r,s\}$ but $S_c^{\mathbf{i},\mathbf{j}} = O, S_c^{\mathbf{i},\mathbf{j}} = O$. To describe this effect one has to consider the directed graph of A instead of the undirected graph of A. In fact, $A_{qr} \neq O$, $A_{rs} = O$, $A_{rq} = O$ and $A_{sr} \neq O$ means that there exists neither the path $\{(q,r),(r,s)\}$ nor the path $\{(s,r),(r,q)\}$.



The block graph of S_c is well–known in graph theory [13], p.11 as the 'edge graph' with respect to the block graph of A.

Example 7.25 *We consider Example 7.20. Here the edges are* $\mathcal{E}_b(A) = \{\{1,2\},\{2,3\},\{4,5\},\{5,6\},\{1,4\},\{2,5\},\{3,6\}\}$. $\mathcal{E}_b(A)$ *is identical to* $\mathfrak{I}$ *and can also be identified with* $\mathcal{V}_b(S_c)$ *using the elements of* $\mathcal{E}_b(A)$ *as labels instead of* $1,2,\ldots,7$. *It easy to see that* $\mathfrak{I}$ *here is identical with* $\mathfrak{I}$ *in Example 7.15.*
From $\mathcal{E}_b(A)$ *we get the following* $\mathcal{P}_p(A)$*:*
$\mathcal{P}_b(A) = \{\ \{\{1,2\},\{1,4\}\},\ \{\{1,2\},\{2,3\}\},\ \{\{1,2\},\{2,5\}\},\ \{\{2,3\},\{2,5\}\},\ \{\{2,3\},\{3,6\}\},$
$\{\{4,5\},\{1,4\}\},\ \{\{4,5\},\{2,5\}\},\ \{\{4,5\},\{5,6\}\},\ \{\{5,6\},\{2,5\}\},\ \{\{5,6\},\{3,6\}\}\ \}$.



Beside this formal graph theoretic derivation of the block graph of S_c from the block graph A there exits another constructive derivation using elimination graphs for LU decomposition [27],pp.93ff. Essentially we can derive the graph of S_c from the fact that S_c is defined as the Schur–complement of the matrix

$$\begin{pmatrix} S & F \\ G & I \end{pmatrix}. \tag{7.26}$$

Definition 7.27 *Let $B = (B_{ij})_{i,j=1,\dots,m}, 1 \leqslant q \leqslant m$. If $G_b(B) = (\mathcal{V}_b(B), \mathcal{E}_b(B))$ is the block graph of B, then the elimination block graph $G_b(B/B_{qq}) = (\mathcal{V}_b(B/B_{qq}), \mathcal{E}_b(B/B_{qq}))$ of B with respect to B_{qq} is defined by*
$\mathcal{V}_b(B/B_{qq}) = \mathcal{V}_b(B) \setminus \{q\}$, $\mathcal{E}_b(B/B_{qq}) = (\mathcal{E}_b(B) \cap (\mathcal{V}_b(B/B_{qq}) \times \mathcal{V}_b(B/B_{qq}))) \cup \mathcal{A}_b(B/B_{qq})$, where $\mathcal{A}_b(B/B_{qq}) = \{\{r,s\} : \{r,q\},\{s,q\} \in \mathcal{E}_b(B)\}$.

The elimination graph can be read as follows. From the initial graph we remove the node q and the corresponding edges $\{q,r\}$ for all r. The remaining graph will get additional edges $\{r,s\}$ for all former adjacent vertices r,s of q. Adjacent vertices of q are those vertices r for which an edge $\{q,r\}$ exists in the old graph.

The main reason for the introduction of elimination graphs is its close relation to the block graph of the Schur–complement

Lemma 7.28 *Let $B = (B_{ij})_{i,j=1,\dots,m}, 1 \leqslant q \leqslant m$ and assume that B_{qq} is nonsingular. Set $C = (B_{r,s})_{r,s=1,\dots q-1,q+1,\dots m} - (B_{r,q})_{r=1,\dots q-1,q+1,\dots m} B_{qq}^{-1} (B_{q,s})_{s=1,\dots q-1,q+1,\dots m}$, which is the Schur–complement of B with respect to B_{qq}.*
Then for the block graph $G_b(C) = (\mathcal{V}_b(C), \mathcal{E}_b(C))$ we have:

$$\mathcal{V}_b(C) = \mathcal{V}_b(B/B_{qq}),\ \mathcal{E}_b(C) \subset \mathcal{E}_b(B/B_{qq}). \tag{7.29}$$

Proof:
This can be obtained from the definition of C: It is clear that $\mathcal{V}_b(C) = \mathcal{V}_b(B/B_{qq})$. Compared with $\mathcal{E}_b((B_{r,s})_{r,s=1,\dots q-1,q+1,\dots m}) = \mathcal{E}_b(B) \cap (\mathcal{V}_b(B/B_{qq}) \times \mathcal{V}_b(B/B_{qq}))$ the block graph of C can have additional edges $\{r,s\}$ only if $C = B_{r,q}B_{qq}^{-1}B_{q,s} \neq O$ or $C = B_{s,q}B_{qq}^{-1}B_{q,r} \neq O$. But this is possible only if the edges $\{r,q\},\{q,r\}$ exist in the block graph of A, i.e. $\{r,s\} \in \mathcal{A}_b(B/B_{qq})$. □

The close relation between *LU* decomposition and graphs is well–known in literature [40] and has been used for several algorithms, e.g. the minimum degree algorithm, see e.g. [40],[18].

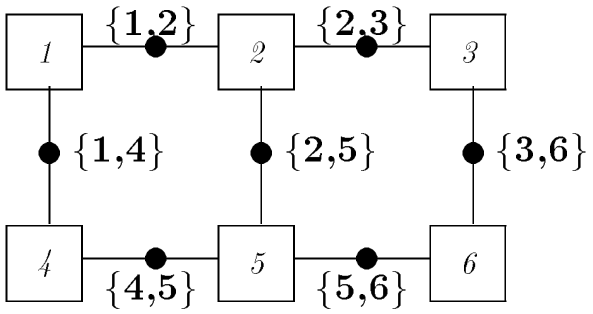
By Lemma 7.28 we have a constructive algorithm to obtain the block graph of S_c. First of all we define the undirected block graph $G_b = (\mathcal{V}_b, \mathcal{E}_b)$ of the augmented matrix from (7.26) by

$$\mathcal{V}_b = \mathcal{V}_b(A) \cup \mathcal{E}_b(A), \mathcal{E}_b = \{\ \{q,\{q,r\}\} : \{q,r\} \in \mathcal{E}_b(A)\}. \tag{7.30}$$

It easy to see that this definition exactly gives the block graph of the augmented matrix from (7.26) in the sense of definition 7.3 up to the use of the edges of $\mathcal{E}_b(A)$ as labels for the additional vertices in $\mathcal{V}_b$.

From the definition of $\mathcal{V}_b, \mathcal{E}_b$ it immediately follows that the block graph of the augmented matrix can be obtained from the block graph of A replacing all edges $\{q,r\}$ by a vertex labelled as $\{q,r\}$ and the two corresponding new edges $\{q,\{q,r\}\},\{q,\{q,s\}\}$

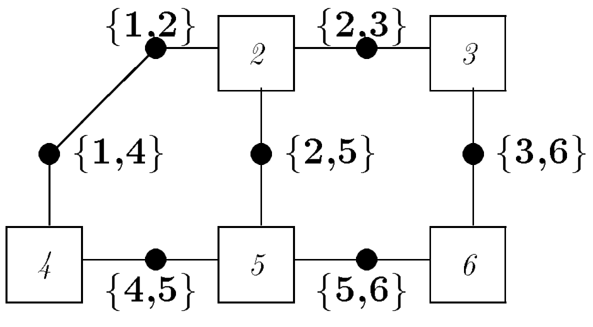
Example 7.31 *We consider example 7.20. According to (7.30) the block graph of the augmented matrix (7.26) looks as follows.*



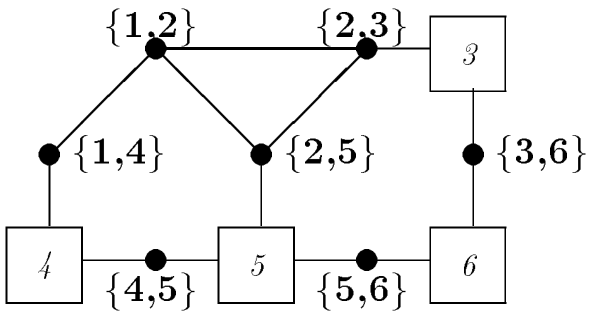
One can see, how the new labels $\{1,2\},\{2,3\},\{4,5\},\{5,6\},\{1,4\},\{2,5\},\{3,6\}\}$ *are just placed as vertices on the previous corresponding edges in the block graph of* A.

Now the block graph of S_c can be constructed from the block graph G_b of the augmented matrix building the elimination graph with respect to the $S_{11},\ldots,S_{pp}$ one after another. Since S is block diagonal, there is no difference in the order of eliminating the vertices $1,2,\ldots,p$.

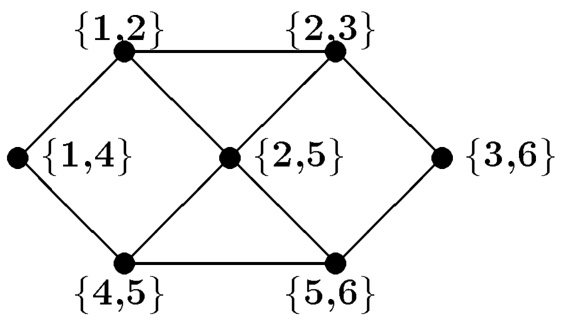
Example 7.32 *We continue Example 7.31 and start eliminating the vertices corresponding to the rectangles. First of all, node* 1 *is removed and its adjacent vertices labelled as* $\{1,2\},\{1,4\}$ *are connected.*



Next we take node 2.



We continue this procedure and finally get the following graph:



In this section we presented two ways of deriving the block graph of S_c from the block graph of A. The investigation of S_c, especially its representation and its block graphs has provided the basis for the following discussion on how the coupling system S_c can be treated in parallel computations.

7.4 Parallel Treatment of S_c

In this section we will discuss how the preception of the representation of S_c in Lemma 7.12 and the relations to the block graph of A from Section 3 can be exploited to get a convenient concept for the parallel treatment of S_c.
From Corollary 7.16 we have got a representation of S_c as

$$S_c = I - \sum_{q=1}^{p} K_q \hat{M}_q K_q^T,$$

where each $\hat{M}_q$ is available on processor q. K_q has been defined in (7.17) by $K_q = (\cdots\ E^{\mathbf{i}}\ \cdots)_{\mathbf{i}\in\mathfrak{I}:\,\mathbf{i}=\{q,r\}}$. Note that q is fixed in the definition of K_q. We set

(7.33) $$K = (K_1, \ldots, K_p)\,.$$

$K_1, \ldots, K_p$ have an interesting property. As shown in Corollary 7.16, any block unit vector $E^{\mathbf{i}} = E^{\{q,r\}}$ from (7.7) appears as block column in K_q and K_r. This induces a natural overlapping distribution for vectors $x = (x^{\mathbf{i}})_{\mathbf{i}\in\mathfrak{I}} \in \mathbb{F}^{n_c}$. In the sequel we will introduce two type of vectors called overlapping type vectors and adding type vectors. The idea of such kind of vectors in the use for parallel computations can be traced back to [55] and has been used in several implementations of finite element methods in parallel computations [45],[46],[6]. In [6] the two type of vectors are referred as consistent and inconsistent vectors.

Definition 7.34 *Let $X \in \mathrm{M}(n_c \times s, \mathbb{R})$ for some number $s > 0$. Then $\overline{X} \in \mathrm{M}(2n_c \times s, \mathbb{R})$ defined by*

(7.35) $$\overline{X} := K^T X$$

is called the **overlapping representation of X**. *Any $\underline{X} \in \mathrm{M}(2n_c \times s, \mathbb{R})$ satisfying*

(7.36) $$K\underline{X} = X$$

is called an **adding representation of X**.

Of course this abstract definition needs some more detailed explanation. In order to avoid confusion about the notation of overlapping and adding type vectors and matrices we like to point out that an **overlapping** representation is **overlined**, otherwise it is underlined. Before we will examine Definition 7.34 and its consequences for the use in parallel computations, we will state some very interesting results related to the use of overlapping type vectors and adding type vectors. Essentially Theorem 7.37 says that we can express all elementary vector operations and matrix–vector operations in terms of overlapping type vectors and adding type vectors. A closer look from the point of parallel computation will be done afterwards.

Theorem 7.37 *Let $X, Y \in \mathrm{M}\,(n_c \times s, \mathbb{R})$ for some number $s > 0$ and let $\overline{X}, \underline{Y} \in \mathrm{M}\,(2n_c \times s, \mathbb{R})$ be corresponding overlapping and adding representations. Using the notation of Lemma 7.12 and Corollary 7.16 the following assertions hold.*

$$\underline{X} = \frac{1}{2}\overline{X} \tag{7.38}$$

is an adding representation of X.

$$\overline{Y} = K^T K \underline{Y} \tag{7.39}$$

is the overlapping representation of Y. Let

$$R = \underline{Y}^T \overline{X}, \tag{7.40}$$

then $R = Y^T X$. Define $\underline{Y}$ by

$$\underline{Y} = \frac{1}{2}\overline{X} - \operatorname{diag}\left(\hat{M}_q\right)_{q=1,\ldots,p} \overline{X}, \tag{7.41}$$

then $\underline{Y}$ is an adding representation of $Y = S_c X$.

Proof:
If $\overline{X} = K^T X$, then by Corollary 7.16 we have $K\frac{1}{2}\overline{X} = \frac{1}{2}KK^T X = X$, which implies (7.38).
$\overline{Y} = K^T K \underline{Y} = K^T Y$, since $\underline{Y}$ is an adding type representation of Y. From this (7.38) follows.
If $K\underline{Y} = Y$, then $Y^T X = \underline{Y}^T K^T X = \underline{Y}^T \overline{X}$.
Finally, by Corollary 7.16 we have

$$S_c = I - \sum_{q=1}^{p} K_q \hat{M}_q K_q^T = I - K \operatorname{diag}(\hat{M}_q)_{q=1,\ldots,p} K^T = K\left(\frac{1}{2}I - \operatorname{diag}(\hat{M}_q)_{q=1,\ldots,p}\right) K^T.$$

Then $Y = S_c X$ becomes $Y = K\left(\frac{1}{2}I - \operatorname{diag}(\hat{M}_q)_{q=1,\ldots,p}\right)\overline{X} = K\underline{Y}$, which shows that $\underline{Y}$ is an adding representation of Y. □

This nice result itself is useless as long as we do not know, which advantages we will have in parallel computations when using these kind of representations.

We will now have a closer look at the definition of overlapping/adding type representations and their meaning. For simplicity let $s = 1$. The case $s > 1$ can be traced back to $s = 1$ by considering any column separately.
Any $x \in \mathbb{F}^{n_c}$ has a natural partitioning $x = (x^{\mathfrak{i}})_{\mathfrak{i} \in \mathfrak{I}}$ taken with respect to (7.7). Since the overlapping representation is defined by $K^T x$ we obtain the vector

$$(7.42) \qquad \overline{x} = \begin{pmatrix} K_1^T x \\ \vdots \\ K_p^T x \end{pmatrix}.$$

Since $K_q = (\cdots \; E^{\mathfrak{i}} \; \cdots)_{\mathfrak{i} \in \mathfrak{I}:\, \mathfrak{i}=\{q,r\}}$, each $K_q^T x$ will consist of $(x^{\{q,r\}})_{r:\{q,r\} \in \mathfrak{I}}$, where q is fixed. From this it follows that any $x^{\{q,r\}}$ will appear twice in $\overline{x}$, one time as part of $K_q^T x$ and another time as part of $K_r^T x$. In the sequel we will assume that $K_q^T x$ is stored on processor q, $q = 1, \dots, p$. We will give an example.

Example 7.43 *We continue Examples 7.15,7.18. Any $x \in \mathbb{F}^{n_c}$ has a natural partitioning as $x = (x^{\mathfrak{i}})_{\mathfrak{i} \in \mathfrak{I}}$. The relation between x and $\overline{x}$ is illustrated in the following table.*

x	$\overline{x}$					
	processor					
	1	2	3	4	5	6
	$K_1^T x$	$K_2^T x$	$K_3^T x$	$K_4^T x$	$K_5^T x$	$K_6^T x$
$x^{\{1,2\}}$	$x^{\{1,2\}}$	$x^{\{1,2\}}$				
$x^{\{2,3\}}$		$x^{\{2,3\}}$	$x^{\{2,3\}}$			
$x^{\{4,5\}}$				$x^{\{4,5\}}$	$x^{\{4,5\}}$	
$x^{\{5,6\}}$					$x^{\{5,6\}}$	$x^{\{6,5\}}$
$x^{\{1,4\}}$	$x^{\{1,4\}}$			$x^{\{1,4\}}$		
$x^{\{2,5\}}$		$x^{\{2,5\}}$			$x^{\{2,5\}}$	
$x^{\{3,6\}}$			$x^{\{3,6\}}$			$x^{\{3,6\}}$

In order to distinguish between the duplicate copies of $x^{\{q,r\}}$ in $\overline{x}$ at two different positions, we denote the two copies of $x^{\{q,r\}}$ by $\overline{x}^{(q,r)}$ and $\overline{x}^{(r,q)}$. This is important, since the two copies of $x^{\{q,r\}}$ are assumed to be stored on different processors! For any $q = 1, \dots, p$ we set

$$(7.44) \qquad \bar{x}^{(q,r)} = \bar{x}^{(r,q)} := x^{\{q,r\}}.$$

Note that $\{q, r\} = \{r, q\}$ but $(q, r) \neq (r, q)$. Then we have

$$(7.45) \qquad K_q^T x = (\bar{x}^{(q,r)})_{r:\{q,r\} \in \mathfrak{I}}, \; \overline{x} = \begin{pmatrix} (\bar{x}^{(1,r)})_{r:\{1,r\} \in \mathfrak{I}} \\ \vdots \\ (\bar{x}^{(p,r)})_{r:\{p,r\} \in \mathfrak{I}} \end{pmatrix}.$$

The assumption that $K_q^T x$ should be stored on processor q now reads as $\overline{x}^{(q,r)}$ is stored on processor q.

We now examine adding type vectors. If $\underline{x} \in \mathbb{F}^{2n_c}$ is an adding type vector, then the equation $K\underline{x} = x$ induces a partitioning for $\underline{x}$. At the first place we have $K = (K_1, \dots, K_p)$ and secondly any K_q can be refined as $K_q = \left(\cdots E^{\{q,r\}} \cdots\right)_{r:\{q,r\}\in\mathfrak{I}}$.
The partitioning of $\underline{x}$ can be done analogously to the partitioning of $\overline{x}$, i.e. we have

$$\underline{x} = (\underline{x}^{(q,r)})_{\{q,r\}\in\mathfrak{I}}. \tag{7.46}$$

$K\underline{x} = x$ now reads as

$$x = \sum_{q=1}^{p} \left(\cdots E^{\{q,r\}} \cdots\right)_{r:\{q,r\}\in\mathfrak{I}} \begin{pmatrix} \vdots \\ \underline{x}^{(q,r)} \\ \vdots \end{pmatrix}_{r:\{q,r\}\in\mathfrak{I}} . \tag{7.47}$$

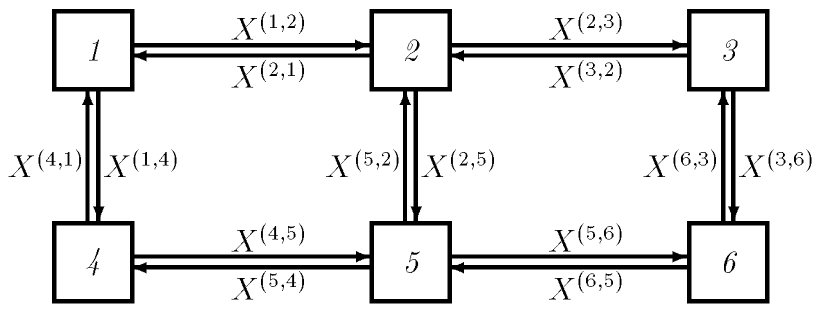
The main reason for the introduction of overlapping and adding type vectors is its convenience in the parallel treatment of S_c. We will show this for the following subroutines which have been mentioned in Theorem 7.37. For these subroutines we will assume that any overlapping and adding representations $\overline{X} = (\overline{X}^{(q,r)})_{\{q,r\}\in\mathfrak{I}}$, $\underline{Y} = (\underline{Y}^{(q,r)})_{\{q,r\}\in\mathfrak{I}}$ of matrices $X, Y \in \mathrm{M}(n_c \times s, \mathbb{R})$ will have their blocks $\overline{X}^{(q,r)}, \underline{X}^{(q,r)}$ stored on processor q.

We can omit a subroutine for (7.38) since the implementation is trivial. For this reason we start with (7.39). From (7.47) it follows that for any $\{q,r\} \in \mathfrak{I}$ we have $X^{\{q,r\}} = \underline{X}^{(q,r)} + \underline{X}^{(r,q)}$.

```
Algorithm 7.48 (adding type → overlapping type from (7.39))
FOR any $\{q,r\} \in \mathfrak{I}$:
    local data exchange $\underline{X}^{(q,r)} \longleftrightarrow \underline{X}^{(r,q)}$ between processor $q, r$.
FOR all $q \in \{1, \dots, p\}$:
    $\overline{X}^{(q,r)} := \underline{X}^{(q,r)} + \underline{X}^{(r,q)}$, for any $r$ such that $\{q,r\} \in \mathfrak{I}$.
```

The local data exchange only affects those pairs q, r of processors such that there is an edge in the block graph of A between q, r. We will illustrate this by an example.

Example 7.49 *Consider the block graph from (7.20), which also corresponds to Example 7.43. The data exchange is illustrated in the following picture.*

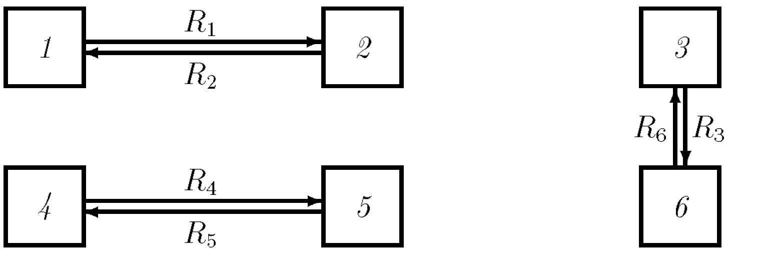


Next we will discuss the realization of (7.40). $R = \underline{Y}^T\overline{X}$ can be written as sum $R = \sum_{q=1}^p R_q$, where any R_q satisfies $R_q = \sum_{r:\{q,r\}\in\Im}(\underline{Y}^{(q,r)})^T\overline{X}^{(q,r)}$.

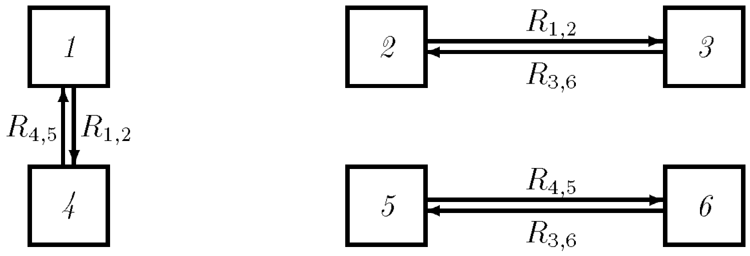
```
**Algorithm 7.50 (scalar product from (7.40))**
FOR all $q \in \{1,\dots,p\}$:
    $R_q = \sum_{r:\{q,r\}\in\Im}(\underline{Y}^{(q,r)})^T\overline{X}^{(q,r)}$
Compute global sum $R = \sum_{q=1}^p R_q$ by data exchange of over all processors.
```

In many realizations of a global sum like $R = \sum_{q=1}^p R_q$ the data exchange and the computation of the sum are combined. We illustrate this by continuing Example 7.49

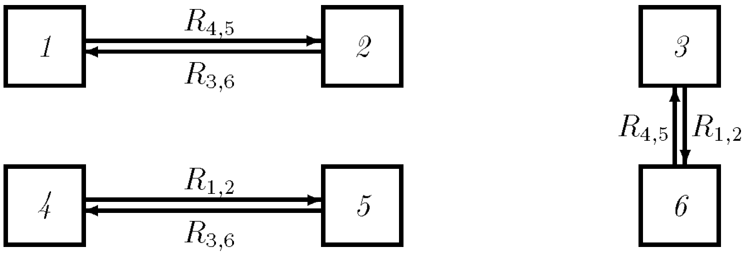
Example 7.51 *Consider Example 7.49. Suppose that $R_1,\dots,R_6$ have already been computed. To compute the sum over all processors we can proceed as follows.*



The computed partial sum we will denote by $R_{1,2}, R_{4,5}$ and $R_{3,6}$. The computation of R can be continued as follows.



As last step we get



The example shows that the computation of a global sum extremely depends on the underlying processor topology, that is the network of data channels between the processors.

Of course both types of communication, local data exchange and global data exchange are dependent on the parallel machine which is used. In addition the local data exchange depends on the preprocessing part which permutes the initial matrix before a system is solved, since one may obtain different connectivity properties for the block graph of A for two different permutations.

As a last step of realization we consider the matrix–vector product from (7.41). The nice result is, that there is no communication necessary, but one starts with the overlapping representation and ends up with an adding representation.

Algorithm 7.52 (matrix–vector product from (7.41))

```
FOR all $q \in \{1, \ldots, p\}$:
        $(\underline{Y}^{(q,r)})_{r:\{q,r\}\in\mathfrak{I}} = \hat{M}_q(\overline{X}^{(q,r)})_{r:\{q,r\}\in\mathfrak{I}}$
        $\underline{Y}^{(q,r)} = \frac{1}{2}\overline{X}^{(q,r)} - \underline{Y}^{(q,r)}$, for any $r$ such that $\{q,r\} \in \mathfrak{I}$.
```

In this section we have introduced a convenient way to treat S_c in parallel computations based on adding type and overlapping type vectors. This concept allows an easy realization of elementary vector operations and matrix–vector operations. Next we will discuss the consequences of this concept for direct and iterative methods applied to S_c.

7.5 Direct Solution of $S_c x = b$

The direct solution of a system $S_c x = b$ requires the explicit generation of S_c and this means that the matrices $\hat{M}_1, \ldots, \hat{M}_p$ from Corollary 7.16 have to be computed explicitly. Although this can be performed in parallel without any communication this might be expensive, if some $n^{\mathbf{i}}, \mathbf{i} \in \mathfrak{I}$ are not small, since for any $q = 1, \ldots, p$ processor q has to solve $\sum_{\mathbf{i}\in\mathfrak{I}:\, q\in\mathbf{i}} n^{\mathbf{i}}$ systems with S_{qq}.
After the generation of $\hat{M}_1, \ldots, \hat{M}_p$ a block LU–decomposition of S_c still has several problems. First of all the matrix S_c is distributed over the processors which means, that the decomposition will be a strongly sequential process. As long as we do not have a stability criterion like positive definiteness, diagonal dominance or the M-matrix property it may happen that we have to do pivoting, which will be rather complicated with respect to the distribution of S_c over the processors. Even if a stability criterion exists, it may happen that one produces fill–in, which requires additional administrational work. A special class where a direct solution method is feasible is the class of acyclic matrices, where fill–in can be avoided, if a stability criterion exists.

Definition 7.53 *Let $B = (B_{ij})_{i,j=1,\ldots,m} \in \mathrm{M}(n \times n, \mathbb{F})$ and let $G_b = (\mathcal{V}_b, \mathcal{E}_b)$ its block graph.*
A sequence of $k > 1$ piecewise disjoint edges $\{q_1, q_2\}, \{q_2, q_3\}, \ldots, \{q_k, q_1\}$ is called **cycle in G_b**.
G_b is called **acyclic**, *if it does not contain any cycle.*

G_b is called a **chain**, *if there exist piecewise disjoint $\{q_1,q_2\},\{q_2,q_3\},\dots,\{q_{m-1},q_m\}$ such that $\mathcal{E}_b \subset \{\{q_1,q_2\},\{q_2,q_3\},\dots,\{q_{m-1},q_m\}\}$.*

Lemma 7.54 *Let $B=(B_{ij})_{i,j=1,\dots,m} \in \mathrm{M}\,(n\times n,\mathbb{F})$. Assume that its block graph is acyclic and that for any permutation matrix P which leaves the blocks B_{ij} invariant, P^TBP has a block LU–decomposition. Then there exists permutation matrix Q such that $Q^TAQ=LU$ is a block LU–decomposition of A and the block graph of L,U is included in the block graph of A.*

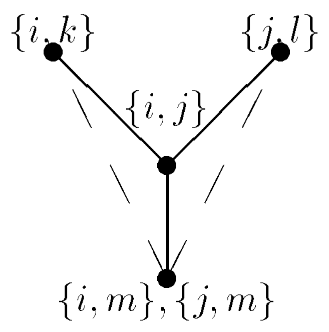
Proof:
See e.g. [31]. □

If a stability criterion exists for S_c and if the block graph of S_c is acyclic, then we can order the blocks of S_c such that we do not produce fill–in. Unfortunately the requirement 'acyclic' for the block graph of S_c is very restrictive. Essentially the only possibility for S_c to have an acyclic block graph is the case when S_c is already block tridiagonal.

Lemma 7.55 *Assume that the block graph of S_c is acyclic and that $\mathcal{E}_b(S_c)=\mathcal{P}_b(A)$ in (7.24), i.e., the block graph of S_c can be derived as in Section 3 and does not have less edges. Then the block graphs of S_c, A are contained in chains.*

Proof:
For the block graph of S_c we use the same labels for the vertices as in Lemma 7.22. Assume that the block graph of S_c is acyclic but not contained in a chain. Then there exists at least one vertex $\mathbf{i}\in\mathcal{V}_b(S_c)$, which has more than two adjacent vertices, i.e., there exist at least three piecewise different vertices $\mathbf{j},\mathbf{k},\mathbf{l}\in\mathcal{V}_b(S_c)\setminus\{\mathbf{i}\}$, each of them labelled as set with two elements, which are adjacent vertices of $\mathbf{i}$. From the special structure of the block graph of S_c described in Lemma 7.22 it follows that we must have nonempty intersections $\mathbf{i}\cap\mathbf{j},\mathbf{i}\cap\mathbf{k},\mathbf{i}\cap\mathbf{l}$. But these three intersections cannot be piecewise disjoint, since $\mathbf{i}$ has only two elements. From this it follows, that at least two of the three vertices $\mathbf{j},\mathbf{k},\mathbf{l}$, say $\mathbf{j},\mathbf{k}$ must have a common intersection with $\mathbf{i}$. Since $\mathbf{j}\neq\mathbf{k}$, it follows that $\mathbf{j}\cap\mathbf{k}=\{q\}$, for some q. For the special block graph of S_c, the edge $\{\mathbf{j},\mathbf{k}\}$ must also belong to $\mathcal{E}_b(S_c)$, since $\mathcal{E}_b(S_c)=\mathcal{P}_b(A)$. So $\mathcal{E}_b(S_c)$ contains the edges $\{\mathbf{i},\mathbf{j}\},\{\mathbf{i},\mathbf{k}\},\{\mathbf{j},\mathbf{k}\}$. This is a cycle and therefore a contradiction to our assumption, that the block graph of S_c is acyclic.
So the block graph of S_c is a contained in a chain, i.e., $\mathcal{V}_b(S_c)=\{\mathbf{i}_1,\dots,\mathbf{i}_s\},\mathcal{E}_b(S_c)\subseteq\{\{\mathbf{i}_1,\mathbf{i}_2\},\{\mathbf{i}_2,\mathbf{i}_3\},\dots,\{\mathbf{i}_{s-1},\mathbf{i}_s\}\}$, where any $\mathbf{i}_k\cap\mathbf{i}_{k+1}$ contains one element. But then $\mathbf{i}_1,\dots,\mathbf{i}_s$ can be written as $\{q_1,q_2\},\{q_2,q_3\},\dots,\{q_s,q_{s+1}\}$, which means that the block graph of A is also a contained in a chain, since $\mathbf{i}_1,\dots,\mathbf{i}_s$ are piecewise disjoint. □

This somehow strange result is illustrated in the following figure, which makes the result more transparent. At least one of the dashed lines must also exist in the picture.



So the case, where A is block tridiagonal is essentially the only case, where the block graph of S_c is acyclic. This case has already been treated in [59].

One way to handle the stability problem as well as the problem with the fill–in in parallel is to collect all $M_1,\dots,M_p$ on one processor or on all processors. Then the problem has been reduced to a usual sequential problem. The disadvantage is of course, that this step requires global communication and unless all $M_1,\dots,M_p$ are small, a lot of data have to be exchanged. Nevertheless, for small $M_1,\dots,M_p$ this strategy is feasible and simplifies the problem extremely.

7.6 Iterative Solution of $S_cx=b$

For the iterative solution of $S_cx=b$, more precisely for the use of Krylov–subspace based methods[74] to solve $S_cx=b$, one only needs elementary operations, like matrix–vector multiplication, scalar products and operations of the form $\alpha x+y$, where $\alpha\in\mathbb{R}$, $x,y\in\mathbb{R}^{n_c}$. Up to the operation $\alpha x+y$ the other two operations were already discussed in Theorem 7.37.

Unfortunately the number of iterations may be huge if the condition number of the eigenvector matrix of S_c is big, or if the eigenvalue distribution is bad[50].

To improve the properties of S_c one usually constructs preconditioners, i.e., nonsingular matrices $\hat S_c$, such that $\hat S_c^{-1}S_c$ has improved properties, but $\hat S_c$ should be relatively cheap to compute and systems of the form $\hat S_cy=c$ should be easy to solve.
The distribution of S_c over the processors and parallel computation aggravate the construction of $\hat S_c$.

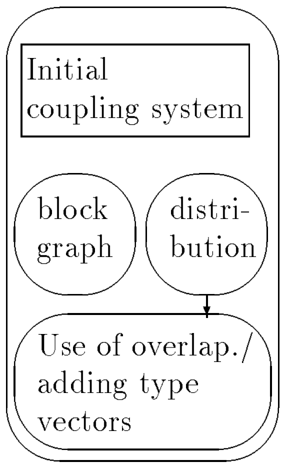
The idea behind the nested Divide & Conquer strategy is to give an additional way of improving the properties of S_c. It is a compromise between a direct solution and an iterative solution of S_c, i.e., the given coupling system is divided into a small part which is directly solved and a remaining reduced coupling system, which still has to be solved. However in principle the nested divide & conquer could be combined with the use of a preconditioner. The only thing which changes is that the updates using orthogonal transformations have to be taken with respect to the preconditioned coupling system instead of the initial coupling system. The problem is how to get a preconditioner for the coupling system and how to parallelize this preconditioner. Even if S_c is explicitly computed any preconditioner for S_c has to take care of the natural distribution of S_c over the processors. By Corollary 7.16

S_c has a natural representation $S_c = I - \sum_{q=1}^{p} K_q \hat{M}_q K_q^T$, where any K_q is describes the blocks of S_c which are located on processor q. A preconditioner which is in natural way adapted to the memory distribution could have the form $\sum_{q=1}^{p} K_q Z_q K_q^T$ since applying this matrix is analogous to applying S_c. In the positive definite case one could choose $\sum_{q=1}^{p} K_q (K_q^T S_c K_q)^{-1} K_q^T$. A general construction of preconditioners of this form will be discussed in future work.

Summary

In this chapter we have discussed the parallel treatment of nested divide & conquer methods. Especially the related coupling system S_c and its representation were of special interest. It has been shown that the coupling system can be written as a sum of p elementary matrices, which overlap only in their block diagonal positions. Each elemental matrix can be generated independently on each processor.
Closely connected to the elementary matrix representation of S_c is its block graph. We have discussed two ways to derive the block graph of S_c from the block graph of the initial system A. First the block graph of S_c is the so–called edge graph with respect to the block graph of A giving a first derivation of the block graph in terms of graph theory. Second we have presented a constructive way using the block elimination graph of a suitably extended system.
The special structure of S_c has turned out to give a convenient parallel treatment using two kind of vectors, overlapping type vectors and adding type vectors. This can be seen as algebraic analogy to techniques which are already used in the parallel treatment of finite element methods for several years. Here analogous results between the relations of overlapping type vectors and adding type vectors and the coupling system could be shown.
For the direct solution we have shown that the most common case of systems where no fill–in is produced during the LU–decomposition, namely the acyclic case, coincides already with the block tridiagonal case. For the iterative solution the nested divide & conquer is a compromise between a direct solution and an iterative solution of S_c.

The coupling system and the topics which we have discussed in this chapter can be summarized in the following picture.



In the next chapter we will generalize this parallel concept to the nested divide & conquer method.

Chapter 8

Parallel Treatment of Nested Divide & Conquer Methods

In this chapter we will discuss the parallel treatment of the nested application of the Sherman–Morrison–Woodbury formula (1.4) from Chapter 3. The concept of parallelization for the initial coupling system from Chapter 7 will be adapted to the nested sequence of coupling systems which is invoked by the nested divide & conquer process.
Again the concept of adding type vectors and overlapping type vectors will be useful to overcome problems which may be caused by using several steps of the nested Sherman–Morrison–Woodbury formula.

8.1 Overview

The aspects which have been discussed in Chapter 7, essentially concentrate on the parallel treatment of S_c. When using the nested divide & conquer approach from Chapter 3, the initial coupling system S_c will be replaced by a sequence of coupling systems $S_{c,k}$. In addition the block diagonal matrix S is replaced by a sequence of matrices S_k, which are typically no longer block diagonal but differ from S up to a low rank matrix. The parallel treatment of this sequence is much more complicated than the initial case, where no nested strategy is applied.
The main idea which we are now going to present is to use the fact that implicitly an LU decomposition is performed on the initial coupling system S_c. Using this fact the sequence of nested coupling systems can be traced back to the initial coupling system with additional low rank updates and pre- and post multiplication with suitable matrices. Of great importance for the parallel realization will be the collection of low rank updates in order to keep the data exchange small. These arguments can be used as well for the representation of S_k^{-1}.

Based on the Nested Divide & Conquer theory in Chapter 3, we now describe its parallel realization. For this we recall the construction of nested splittings in (3.4)–(3.8). For a given initial splitting $A = S - FG$ from (7.10) the nested strategy can be described as follows: We consider numbers $0 < s_0, \dots, s_{m-1}$ with $\sum_{k=0}^{m-1} s_k < n_c$. Furthermore we set $r_0 = n_c$, $r_{k+1} = r_k - s_k$, $k = 0, \dots, m-1$. Using these settings we have considered for any $k = 0, \dots, m-1$ orthogonal matrices $U_k \in \mathrm{GL}\,(r_k, \mathbb{F})$ partitioned as $U_k = \left[\tilde{U}_k, \hat{U}_k\right]$, where $\tilde{U}_k \in \mathrm{M}\,(r_k \times s_k, \mathbb{R})$, $\hat{U}_k \in \mathrm{M}\,(r_k \times r_{k+1}, \mathbb{R})$. The numbers $s_0, \dots, s_{m-1}$ are assumed to be small compared with n_c.

By the aid of these matrices the nested sequence of splittings is defined by $S_0 := S, F_0 := F, G_0 := G$ and for $k = 0, \dots, m-1$:

$$\begin{aligned} \left[\tilde{F}_k,\ F_{k+1}\right] &:= F_k U_k, \\ \left[\begin{smallmatrix} \tilde{G}_k \\ G_{k+1} \end{smallmatrix}\right] &:= U_k^T G_k, \\ S_{k+1} &:= S_k - \tilde{F}_k \tilde{G}_k. \end{aligned} \tag{8.1}$$

Since $S_{k+1} = S_k - \tilde{F}_k \tilde{G}_k$, $k = 0, \dots, m-1$ and $A = S_k - F_k G_k$, $k = 0, \dots, m$ we can express S_{k+1}^{-1}, A^{-1} using the Sherman–Morrison–Woodbury formula (1.4).

$$\Longrightarrow \left\{ \begin{array}{rcl} S_{k+1}^{-1} &=& S_k^{-1} + S_k^{-1} \tilde{F}_k \tilde{S}_{c,k}^{-1} \tilde{G}_k S_k^{-1}, \\ A^{-1} &=& S_{k+1}^{-1} + S_{k+1}^{-1} F_{k+1} S_{c,k+1}^{-1} G_{k+1} S_{k+1}^{-1}. \end{array} \right. \tag{8.2}$$

Here we have set

$$\tilde{S}_{c,k} := I - \tilde{G}_k S_k^{-1} \tilde{F}_k,\ S_{c,k+1} := I - G_{k+1} S_{k+1}^{-1} F_{k+1}. \tag{8.3}$$

In (3.13) we have introduced in addition

$$Y_k = \hat{U}_0(\cdots \hat{U}_{k-2}(\hat{U}_{k-1}\tilde{U}_k)\cdots). \tag{8.4}$$

Using Y_k we can rewrite $\tilde{F}_k$ as $\tilde{F}_k = F Y_k$. A further matrix $E_k = S_k^{-1}\tilde{F}_k$ has been defined in order to simplify the use of $S_k^{-1}\tilde{F}_k$ for $\tilde{S}_{c,k}$ and S_{k+1}^{-1}. Once E_k has been computed we obtain

$$\tilde{S}_{c,k} = I - \tilde{G}_k E_k,\ S_{k+1}^{-1} = (I + E_k \tilde{S}_{c,k}^{-1} \tilde{G}_k) S_k^{-1}. \tag{8.5}$$

This has lead to product representation

$$S_{k+1}^{-1} = (I + E_k \tilde{S}_{c,k}^{-1} \tilde{G}_k) \cdots (I + E_0 \tilde{S}_{c,0}^{-1} \tilde{G}_0) S^{-1}, \tag{8.6}$$

as it was shown in (3.11).

In this nested definition we have assumed that all S_k are nonsingular. In this case the corresponding coupling systems are nonsingular, too.

For the practical implementation of course not all matrices have to be computed explicitly. Since in addition we would like to implement this method on a parallel machine we have to look closer at this sequence of matrices.
Applying S_{k+1}^{-1} to a vector b means that we have to solve a system $Sx = b$ followed by several low rank updates. From the product representation of S_{k+1}^{-1} in (8.6) it follows that

we have to compute $k+1$ scalar products and vector updates of the form $\alpha x + y$ one after another. Since the vector and the matrix are assumed to be distributed over the processors, this will require $k+1$ steps of global communication. The solution process will be illustrated in the following tabular.

Step	Operation	Communication
	$b = S^{-1}b$	—
0	$b = b + E_0\tilde{S}_{c,0}^{-1}\tilde{G}_0 b$	global. comm. for $R = \tilde{G}_0 b$
1	$b = b + E_1\tilde{S}_{c,1}^{-1}\tilde{G}_1 b$	global. comm. for $R = \tilde{G}_1 b$
	$\vdots$	
k	$b = b + E_k\tilde{S}_{c,k}^{-1}\tilde{G}_k b$	global. comm. for $R = \tilde{G}_k b$

The situation will be quite similar for $S_{c,k+1}$, since S_{k+1}^{-1} is part of it. When using Householder reflectors for the orthogonal matrices U_k the same problem will occur.
For this reason, we will discuss the following topics:

- The general treatment of a product of low rank modification matrices $I - VB^{-1}W$ in parallel computations.
- The treatment of $U_0, \ldots, U_{m-1}$, especially the way to handle a product of Householder reflectors and how Householder reflectors can be used in combination with vectors of adding type and overlapping type from Chapter 7.
- The parallel treatment of $S_{c,k+1}$ from (8.3).
- The realization of the product representation of S_{k+1}^{-1} from (8.6).

The templates used for the parallel solution of $S_{c,m}x = b$ are summarized in the following table.

Algorithm	Purpose	Subject
8.10	Compute Householder reflector of distributed vector	Householder reflectors
8.15	Form product of Householder reflectors	
8.16 8.17	Apply Householder reflectors	
8.20	Matrix–vector product $S_{c,m}x$	Handle $S_{c,m}$
8.23	Update vectors $B_{m-1,0}, \ldots, B_{m-1,m-2}$ from LU–decomposition of S_c	
8.25	Form product of Householder reflectors, version adapted for the use with nested divide & conquer	
8.26	New low rank update from step $m-1 \to m$	
8.27	Parallel nested divide & conquer, application of the templates to treat $S_{c,m}x = b$ in parallel	
8.29	Multiplication with first m columns from the product of Householder reflectors	Handle S_m^{-1}
8.30	Multiplication with first m columns from the product of Householder reflectors	
8.31	Solve a system $S_m x = b$ in parallel	

8.2 Treatment of Products of Low Rank Modifications

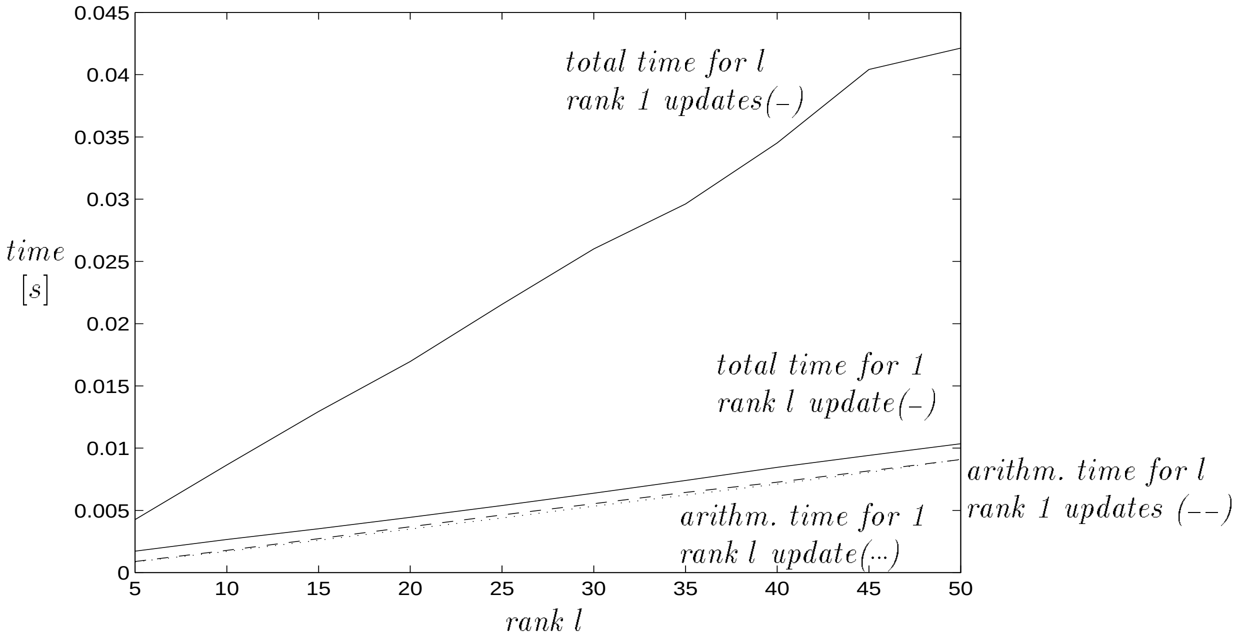
As a first step to get a sensible parallel realization of nested splittings obtained by the divide & conquer approach from Chapter 3, we will discuss how a product of the form

$$(I - V_1 B_1^{-1} W_1) \cdots (I - V_k B_k^{-1} W_k)$$

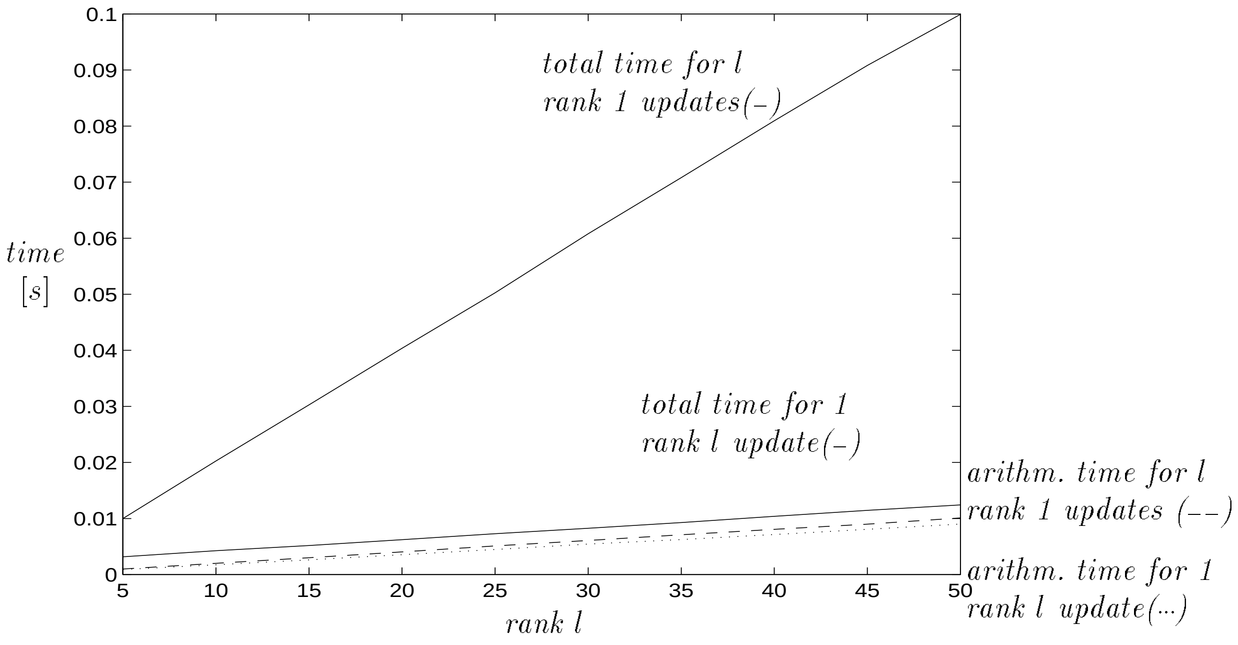
can be modified in order to be effective in parallel computations. A product of this form occurs twice when applying the nested Sherman–Morrison–Woodbury formula from (1.4). At the first place it appears in the product representation of S_{k+1}^{-1} in (8.6) and secondly the product of orthogonal matrices $U_0, \ldots, U_{m-1}$ will be of this form when using their Householder representation.

The product $(I - V_1 B_1^{-1} W_1) \cdots (I - V_k B_k^{-1} W_k)$ itself applied to a vector x requires k scalar products like $R_k = W_k x$ and additional updates like $x - V_k B_k^{-1} R_k$. For vectors which are distributed over the processors any scalar product will require global communication as we have already discussed in Algorithm 7.50. For practical purposes, communication depends not strongly linear on the length of the data, but it depends on one hand on a fixed latency time which is needed to set up communication and on the other hand it depends on the length of the data. For small number of data the second part is almost neglectible. Thus it is typically many times cheaper to exchange a block of data in one step than to exchange one value several times.

Example 8.7 (Computation Time for Low Rank Updates) *We will illustrate this by a practicle example. We compare for rank 1 updates of size $n \times n$, where $n = 1000$, the time which is requested by a product of say l rank 1 updates compared with the time which is needed by a rank l update. The computations were carried out on a PARSYTEC GCPP-128. This is a MIMD parallel computer with a distributed memory architecture. For the numerical experiments we used $p = 4$ processors. For the size l of the rank we used $l = 5, 10, \ldots, 50$. In the following picture we compared the total time as well as the maximum time for arithmetic operations for both variants. To have reliable results, the operations were carried out 10000 times and the average was taken over the computational time.*
We can see in the following picture that for the product of l rank 1 updates as well as for the rank l update the maximum number of arithmetic time is almost the same and it grows linearly with the rank, which is expected. A great difference is the total time, or more precisely the communication time. For the rank l update the communication time is almost constant, while for the product of l rank 1 updates the communication times grows rapidly. Even for small l, e.g. $l = 10$ the communication for the product of rank 1 updates is more than three times as much as the time for the arithmetic operations.



The situation becomes more dramatic, if we increase the size of the problem, e.g. $n = 4000$*,* $p = 16$*. For larger number of processors, the time which is needed for the global communication will increase while the arithmetic time will be almost the same as before.*



The example illustrates that handling of a product of low rank updates is a serious problem, if the data traffic should not overlay the computation.
This observation is the background for the following lemma, which describes how a product of low rank modifications can be collected to one matrix.

Lemma 8.8 *Consider for* $l = 1, \ldots, k$*,* $V_l, W_l^T \in \mathrm{M}\,(n \times n_l, \mathbb{F})$*,* $B_l \in \mathrm{GL}\,(n_l, \mathbb{F})$*. Then*

$$(I - V_1 B_1^{-1} W_1) \cdots (I - V_k B_k^{-1} W_k) = I - V B^{-1} W,$$

where

$$V=[V_1,\dots,V_k],\ B=\begin{pmatrix} B_1 & W_1V_2 & \cdots & W_1V_k \\ & \ddots & \ddots & \vdots \\ & & \ddots & W_{k-1}V_k \\ 0 & & & B_k \end{pmatrix},\ W=\begin{bmatrix} W_1 \\ \vdots \\ W_k \end{bmatrix}.$$

Proof:
This follows immediately by induction on k. □

Applying Lemma 8.8 to a product of low rank modifications $(I - V_1B_1^{-1}W_1)\cdots(I - V_kB_k^{-1}W_k)$ requires the additional computation of W_iV_l. This will be an additional overhead. But if some columns of this product are explicitly needed, then this overhead will be compensated by the fact that we can easily compute a column of $I - VB^{-1}W$.

$$(I - VB^{-1}W)e_k = e_k - VB^{-1}w_k.$$

Here e_k should denote the k–th unit vector and w_k denotes column k of W.

8.3 Handling a Product of Householder Reflectors

Now we will discuss the treatment of $U_0,\dots,U_{m-1}$ or more precisely the treatment of the product

(8.9) $$Q_k := U_0\begin{pmatrix} I & 0 \\ 0 & U_1 \end{pmatrix}\cdots\begin{pmatrix} I & 0 \\ 0 & U_k \end{pmatrix}.$$

For the realization we have to perform several multiplications with $U_0,\dots,U_k$. We will assume, that all $U_0,\dots,U_k$ are represented by a product of Householder reflectors[41], i.e. products of matrices of the form

$$I - \beta vv^T, v \in \mathbb{R}^n.$$

Before we discuss this in detail we give an algorithm which solves the problem

$$(I - \beta vv^T)u = \sigma e_k$$

for a given vector $u \in \mathbb{R}^n$ in overlapping representation $\overline{u}$. Here e_k denotes the k–th unit vector. Since $I - \beta vv^T$ is assumed to be orthogonal, σ is unique up to a sign. Following [41], p.196, we can adapt the computation to the parallel case. This problem has been studied in [76],[77]. Here we will concentrate on the adaption to the use of adding type vectors and overlapping type vectors.

```
Algorithm 8.10 (Compute Householder reflector)
FOR all $q \in \{1,\dots,p\}$.
        Compute local scalar product of $\rho = u^T u$:
        $\underline{u}_q := \frac{1}{2}\overline{u}_q$. $\rho_q = \underline{u}_q^T \overline{u}_q$.
        Get component $k$ of distributed stored vector $u$:
        IF   $\overline{u}_q$ contains the $k$–th component of $u$, then let $\nu_q$ be this element.
        ELSE $\nu_q = 0$.
Compute global sum $[\rho,\nu] = \sum_{q=1}^{p}[\rho_q,\nu_q]$ by data exchange of over all processors.
$\rho = \sqrt{\rho}$, $\nu = \frac{1}{2}\nu$.
Compute Householder parameters:
IF $\mu \neq 0$, $\alpha = \nu + \mu \operatorname{sgn}\nu$, $v = u/\alpha$, $\beta = -|\alpha|/\mu$.
IF $\overline{v}_q$ contains the $k$–th component of $v$ then set this component to 1.
```

We can immediately apply Lemma 8.8 to a product of Householder reflectors $(I - \beta_0 v_0 v_0^T)\cdots(I - \beta_s v_s v_s^T)$. This requires the additional computation of $v_i^T v_l$. This will be an additional overhead. In our case we will see that these products have to be performed in any case, even if we keep the reflectors in factored form. The reason for this is the fact that we need some columns of the product Q_k of reflectors explicitly. Y_k defined in (8.4) is essentially k–th column of Q_k, if $s_0 = s_1 = \cdots = s_k$. Whenever we increase the number of Householder reflectors from k to $k+1$, we have to compute $v_0^T v_{k+1}, \dots, v_k^T v_{k+1}$. Again the communication can be done in one step exchanging k values.
For the parallel treatment of S_c we will see that multiplications with one or a block of Householder transformations can be easily performed based on adding type vectors and overlapping type vectors.

Lemma 8.11 *Let $x, y \in \mathbb{R}^{n_c}$ and denote by $\overline{x}, \underline{y} \in \mathbb{R}^{2n_c}$ corresponding overlapping and adding representations. Let $V, W \in \mathrm{M}(n_c \times k, \mathbb{R}), T \in \mathrm{GL}(k, \mathbb{R})$ and denote by $\overline{V}, \overline{W} \in \mathrm{M}((2n_c) \times k, \mathbb{R})$ corresponding overlapping representations of V, W. Then*

$$(I - \frac{1}{2}\overline{V}T^{-1}\overline{W}^T)\overline{x} \tag{8.12}$$

is the overlapping representation of $(I - VT^{-1}W^T)x$,

$$(I - \frac{1}{2}\overline{V}T^{-1}\overline{W}^T)\underline{y} \tag{8.13}$$

is an adding representation of $(I - VT^{-1}W^T)y$.

Proof:
For (8.12) we have to show that $(I - \frac{1}{2}\overline{V}T^{-1}\overline{W}^T)\overline{x} = K^T(I - VT^{-1}W^T)x$. But $K^T(I - VT^{-1}W^T)x = \overline{x} - K^T V T^{-1} W^T \frac{KK^T}{2} x = \overline{x} - \frac{1}{2}\overline{V}T^{-1}\overline{W}^T\overline{x}$.
For (8.13) we have to show that $K(I - \frac{1}{2}\overline{V}T^{-1}\overline{W}^T)\underline{y} = (I - VT^{-1}W^T)y$. In this case we have $K(I - \frac{1}{2}\overline{V}T^{-1}\overline{W}^T)\underline{y} = y - \frac{1}{2}KK^T V T^{-1} W^T K\underline{y} = y - VT^{-1}W^T y$. □

If we have collected a product of Householder matrices to one matrix according to Lemma 8.8, then by Lemma 8.11 we need the overlapping representation of this matrix in order to apply it to both kind of vectors, adding type vectors and overlapping type vectors. This will be the case for Householder reflectors.

Analogously to Theorem 7.37 and the related Algorithms 7.48, 7.50,7.52 we will formulate subroutines to compute the collected Householder representations and to apply the collected product to a vector. For these subroutines we will assume that $v_0, \ldots, v_s, v_{s+1}$ are given by their overlapping representations. Any overlapping and adding representation $\overline{x} = (\overline{x}^{(q,r)})_{\{q,r\}\in\mathfrak{I}}$, $\underline{y} = (\underline{y}^{(q,r)})_{\{q,r\}\in\mathfrak{I}}$ will have its blocks $\overline{x}^{(q,r)}, \underline{x}^{(q,r)}$ stored on processor q.

We start with the collection of Householder reflectors from Lemma 8.8. According to Lemma 8.8 suppose that $(I - \beta_0 v_0 v_0^T) \cdots (I - \beta_s v_s v_s^T)$ have already been collected to one matrix

$$(I - \beta_0 v_0 v_0^T) \cdots (I - \beta_s v_s v_s^T) = I - V_s T_s^{-1} V_s^T,$$

where

$$V_s = [v_0, \ldots, v_s]\,,\; T_s = (T_{ij})_{i,j=0,\ldots,s} = \begin{pmatrix} \frac{1}{\beta_0} & v_0^T v_1 & \cdots & v_0^T v_s \\ & \ddots & \ddots & \vdots \\ & & \ddots & v_{s-1}^T v_s \\ \text{O} & & & \frac{1}{\beta_s} \end{pmatrix}. \tag{8.14}$$

Then the product $(I - \beta_0 v_0 v_0^T) \cdots (I - \beta_s v_s v_s^T)(I - \beta_{s+1} v_{s+1} v_{s+1}^T) = I - V_{s+1} T_{s+1}^{-1} V_{s+1}^T$ can be computed by the following algorithm.

Algorithm 8.15 (Product of Householder reflectors from Lemma 8.8)
Let $V \equiv V_s, v \equiv v_{s+1}$ *and compute* $R = v^T V$ *by calling Algorithm 7.50 with* $\overline{X} = \overline{V}$ *and* $\underline{Y} = \frac{1}{2}\overline{v}$.
$\Rightarrow [T_{s+1,1}, \ldots, T_{s+1,s}] = R,\; T_{s+1,s+1} = 1/\beta_{s+1}$.

If the product of Householder reflectors is given by $I - V_s T_s^{-1} V_s^T$ then the following algorithms compute $\Psi = (I - V_s T_s^{-1} V_s^T)\Phi$ for $\Phi \in \mathrm{M}(n_c, \times s, \mathbb{R})$. In order to get no conflict in the use of X, Y in Algorithm 7.50, the matrices here are called Φ and Ψ. The first algorithm is used for overlapping type vectors and the second one is used for adding type vectors. In practice, the algorithms coincide, i.e., one can apply any of the two Algorithms 8.16,8.17 to both kind of vectors.

Algorithm 8.16 (Apply Householder reflectors from Lemma 8.11)
Let $V \equiv V_s, T \equiv T_s$ *and compute* $R = \Phi^T V$ *by calling Algorithm 7.50 with* $\overline{X} = \overline{V}$ *and* $\underline{Y} = \frac{1}{2}\overline{\Phi}$.

```
Solve $T\hat{R} = R$ simultaneously on all processors.
FOR all $q \in \{1, \ldots, p\}$:
    $\overline{\Psi}^{(q,r)} = \overline{\Phi}^{(q,r)} - \overline{V}^{(q,r)}\hat{R}$ for any $r$ such $\{q,r\} \in \mathfrak{I}$.
```

Algorithm 8.17 (Apply Householder reflectors from Lemma 8.11)

```
Let $V \equiv V_s, T \equiv T_s$ and compute $R = \Phi^T V$ by calling Algorithm 7.50 with $\overline{X} = \overline{V}$ and $\underline{Y} = \underline{\Phi}$.
Solve $T\hat{R} = R$ simultaneously on all processors.
FOR all $q \in \{1, \dots, p\}$:
    $\underline{\Psi}^{(q,r)} = \underline{\Phi}^{(q,r)} - \frac{1}{2}\overline{V}^{(q,r)}\hat{R}$ for any $r$ such $\{q,r\} \in \Im$.
```

8.4 Parallel Treatment of $S_{c,m}$

After the parallel treatment of Householder reflectors we will discuss the way how the coupling system $S_{c,m}$ from (8.3) which is generated by the nested divide & conquer approach from Chapter 3 can be treated in parallel computations. For $m = 0$ this was already the topic of Chapter 7. When introducing low rank updates the remaining coupling system $S_c \equiv S_{c,0}$ will be replaced by its Schur–complement, after an equivalence transformation with U_0. This has been shown in Lemma 3.18. This gives a way to treat the coupling system $S_{c,m}$ from (3.7), which is involved by the nested divide & conquer process, in parallel. From Lemma 3.18 it follows that we can write for $k = 0, \dots, m-1$, $\tilde{S}_{c,k}$ as

$$\tilde{S}_{c,k} = \tilde{U}_k^T \hat{U}_{k-1}^T \cdots \hat{U}_0^T S_c \hat{U}_0 \cdots \hat{U}_{k-1} \tilde{U}_k - \sum_{l=0}^{k-1} \tilde{B}_{kl} \tilde{S}_{c,l}^{-1} \tilde{B}_{lk} \tag{8.18}$$

and $S_{c,m}$ as

$$S_{c,m} = \hat{U}_{m-1}^T \cdots \hat{U}_0^T S_c \hat{U}_0 \cdots \hat{U}_{m-1} - \sum_{l=0}^{m-1} B_{ml} \tilde{S}_{c,l}^{-1} B_{lm}. \tag{8.19}$$

Before we will describe the matrices $\tilde{B}_{kl}, B_{lm}$ let us assume for a moment that these matrices are available. Essentially (8.19) says that we have to apply the product of Householder reflectors and its transpose to S_c followed by m additional low rank modification to obtain $S_{c,m}$. We can state an algorithm for a multiplication $y = S_{c,m}x$. For this algorithm, we assume that B_{ml}, B_{lm} are given by their adding representation, $\tilde{S}_{c,l}$ should be globally available. The vector x is assumed to be given by its overlapping representation analogously to the standard case when $y = S_c x$ is computed. Since the size of $S_{c,m}$ is less than the size of S_c we assume that x, y are extended by $n_c - r_m$ leading zeros.

```
Algorithm 8.20 (Matrix–Vector Product $S_{c,m}x$)
Let $B \equiv [B_{m-1,0},\dots,B_{m,m-1}]$, $B' \equiv \begin{bmatrix} B_{0,m-1} \\ \vdots \\ B_{m-1,m} \end{bmatrix}$, $\tilde{S}_c \equiv \operatorname{diag}(\tilde{S}_{c,0},\dots,\tilde{S}_{c,m-1})$,
Apply Algorithm 8.16 with $\overline{\Phi} = \overline{x}$ and result $\overline{\Psi}$.
FOR all $q \in \{1,\dots,p\}$.
        Compute local scalar product of $R = x^T B$:
        $R_q = \underline{B}_q^T \overline{x}_q$.
Compute global sum $R = \sum_{q=1}^{p} R_q$ by data exchange of over all processors.
This sum can be build simultaneously with the global sum in Algorithm 8.16.
FOR all $q \in \{1,\dots,p\}$.
        $\underline{y}^{(q,r)} = \underline{B}^{(q,r)}\tilde{S}_c^{-1}R$ for any $r$ such $\{q,r\} \in \mathfrak{I}$.
Apply Algorithm 7.52 with $\overline{X} := \overline{\Psi}$ and result $\underline{Y}$.
Apply Algorithm 8.17 with $\underline{\Phi} = \underline{Y}$ and result $\underline{\Psi}$.
FOR all $q \in \{1,\dots,p\}$.
        $\underline{y}^{(q,r)} = \underline{y}^{(q,r)} + \underline{\Psi}^{(q,r)}$ for any $r$ such $\{q,r\} \in \mathfrak{I}$.
```

We have seen that the introduction of B_{lm}, B_{ml} only requires two steps of global communication, while the usual application of $S_{c,m} = I - G_m S_m^{-1} F$ requires $3*(m-1)$ steps of global communication, $m-1$ steps for S_m^{-1} and twice application of Householder reflectors requires $2(m-1)$ steps of global communication. For the collected Householder product the usual application of $S_{c,m}$ would still require $m+1$ steps of global communication.

We now study the matrices $\tilde{B}_{kl}$ and B_{lm}. Following (3.22), $\tilde{B}_{kl}$ and $\tilde{B}_{lk}$, $0 \leqslant l < k < m$, are defined by

$$\tilde{B}_{kl} = \hat{U}_k^T \hat{U}_{k-1}^T \cdots \hat{U}_l^T S_{c,l} \tilde{U}_l, \; \tilde{B}_{lk} = \tilde{U}_l^T S_{c,l} \hat{U}_l^T \cdots \hat{U}_{k-1} \tilde{U}_k. \tag{8.21}$$

According to (3.23) B_{mk} and B_{km}, $k = 0,\dots,m-1$ satisfy

$$B_{mk} = \hat{U}_{m-1}^T \cdots \hat{U}_k^T S_{c,k} \tilde{U}_k, \; B_{km} = \tilde{U}_k^T S_{c,k} \hat{U}_k^T \cdots \hat{U}_{m-1}. \tag{8.22}$$

The definition of $\tilde{B}_{kl}$, $\tilde{B}_{lk}$, B_{mk} and B_{km} can be made more transparent, when these matrices are successively generated and updated. We will show this by the following (sequential) scheme.

$m = 1$: Set $B_{1,0} = \hat{U}_0^T S_{c,0} \tilde{U}_0$, $B_{0,1} = \tilde{U}_0^T S_{c,0} \hat{U}_0$, $\tilde{S}_{c,0} = \tilde{U}_0^T S_c \tilde{U}_0$.
$m > 1$: Assume that for all $k = 0,\dots,m-2$, $B_{m-1,k}$ and $B_{k,m-1}$, have been computed.

FOR $k = 0,\dots,m-2$:

$$\begin{bmatrix} \tilde{B}_{m-1,k} \\ B_{m,k} \end{bmatrix} := \begin{bmatrix} \tilde{U}_{m-1}^T \\ \hat{U}_{m-1}^T \end{bmatrix} B_{m-1,k}, \; \left[\tilde{B}_{k,m-1}, B_{k,m}\right] := B_{k,m-1}\left[\tilde{U}_{m-1}, \hat{U}_{m-1}\right].$$

$B_{m,m-1} = \hat{U}_{m-1}^T S_{c,m-1} \tilde{U}_{m-1}$, $B_{m-1,m} = \tilde{U}_{m-1}^T S_{c,m-1} \hat{U}_{m-1}$.
$\tilde{S}_{c,m-1} = \tilde{U}_{m-1}^T S_{c,m-1} \tilde{U}_{m-1}$.

This scheme shows that in any step from $m-1$ to m the old matrices $B_{k,m-1}, B_{m-1,k}$ will be updated by a Householder transformation to become $\tilde{B}_{m-1,k}$, $B_{m,k}$ and $\tilde{B}_{k,m-1}$, $B_{k,m}$. Then new matrices $B_{m,m-1}$, $B_{m-1,m}$ are generated. Of course for the use in parallel computations the computation of $B_{m,m-1}$, $B_{m-1,m}$ has to make use of the representation of $S_{c,m-1}$ from (8.19) with m replaced by $m-1$. In this case the computation of $B_{m,m-1}$ and $B_{m-1,m}$ changes to the following sequence.

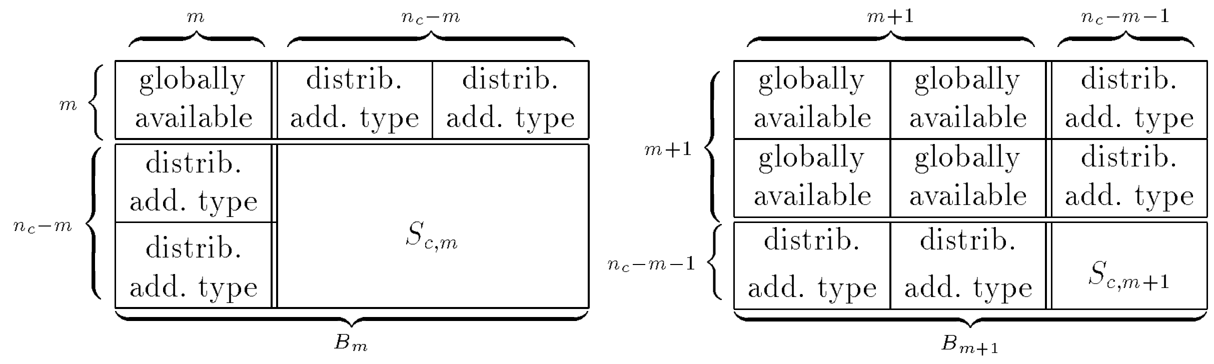
$$\begin{aligned}
Y_{m-1} &:= \hat{U}_0 \cdots \hat{U}_{m-2}\tilde{U}_{m-1}.\\
B_{m,m-1} &:= \hat{U}_{m-1}^T \hat{U}_{m-2}^T \cdots \hat{U}_0^T S_c Y_{m-1} - \textstyle\sum_{l=0}^{m-2} B_{m,l}\tilde{S}_{c,l}^{-1}\tilde{B}_{l,m-1}.\\
B_{m-1,m} &:= Y_{m-1}^T S_c \hat{U}_0 \cdots \hat{U}_{m-2}\hat{U}_{m-1} - \textstyle\sum_{l=0}^{m-2} \tilde{B}_{m-1,l}\tilde{S}_{c,l}^{-1} B_{l,m}.\\
\tilde{S}_{c,m-1} &:= Y_{m-1}^T S_c Y_{m-1} - \textstyle\sum_{l=0}^{m-2} \tilde{B}_{m-1,l}\tilde{S}_{c,l}^{-1}\tilde{B}_{l,m-1}.
\end{aligned}$$

For a sensible use of $\tilde{B}_{kl}$, $\tilde{B}_{lk}$, B_{mk} and B_{km} we will now present a concept. These additional low rank modifications have to be sensibly handled in order to avoid too much additional communication. The concept can be divided into three parts. The first part will be a strategy for the distribution of $\tilde{B}_{kl}$, $\tilde{B}_{lk}$, B_{mk} and B_{km}. The second part will be a analogous strategy for the distribution of V_s from (8.14). Finally the third part will consider the computation of $B_{m-1,m}$, $B_{m,m-1}$ and $\tilde{S}_{c,m-1}$.

To present the first part we set

$$B_m = \left(\begin{array}{cccc||c}
\tilde{S}_{c,0} & \tilde{B}_{0,1} & \cdots & \tilde{B}_{m-1,0} & B_{0,m}\\
\tilde{B}_{1,0} & \ddots & \ddots & \vdots & \vdots\\
\vdots & \ddots & \ddots & \tilde{B}_{m-2,m-1} & B_{m-2,m}\\
\tilde{B}_{m-1,0} & \cdots & \tilde{B}_{m-1,m-2} & \tilde{S}_{c,m-1} & B_{m-1,m}\\
\hline\hline
B_{m,0} & \cdots & B_{m,m-2} & B_{m,m-1} & S_{c,m}
\end{array}\right).$$

The idea is now to keep some part of B_m, namely the upper left part with $\tilde{S}_{c,k}$, $\tilde{B}_{kl}$, $\tilde{B}_{lk}$ on all processors. Since $s_0, \ldots, s_{m-1}$ are small compared with n_c, this requires not too much storage and overhead. The part with B_{lm}, B_{ml} will be held as adding type vectors, distributed over the processors. Finally $S_{c,m}$ will stay in its factored form as in (8.19). We give a sketch of the memory distribution for B_m and B_{m+1} for the case $s_0 = \cdots = s_m = 1$:



In order to provide this memory distribution we will give an algorithm which computes $\begin{bmatrix} \tilde{B}_{m-1,0} & \cdots & \tilde{B}_{m-1,m-2}\\ B_{m,0} & \cdots & B_{m,m-2}\end{bmatrix}$ from $B_{m-1,0}, \ldots, B_{m-1,m-2}$ and which makes $\tilde{B}_{m-1,0}, \ldots, \tilde{B}_{m-1,m-2}$ globally available. For the algorithm we assume that $B_{m-1,0}, \ldots, B_{m-1,m-2}$ are expanded by $m-2$ leading zeros and that $B_{m-1,0}, \ldots, B_{m-1,m-2}$

are given by their adding representations. We assume that U_{m-1} is given by its Householder representation $= I - \beta_{m-1} v_{m-1} v_{m-1}^T$, computed by Algorithm 8.10.

```
Algorithm 8.23 (Update $B_{m-1,0}, \ldots, B_{m-1,m-2}$)
Let $B \equiv [B_{m-1,0}, \ldots, B_{m-1,m-2}]$, $v \equiv v_{m-1}$, $\beta \equiv \beta_{m-1}$.
FOR all $q \in \{1, \ldots, p\}$.
        Compute local scalar product of $R = v^T B$:
        $\underline{R}_q = \overline{v}_q^T B$.
        Get row $m-1$ of distributed stored matrix $B$:
        IF    $\underline{B}_q$ contains the $(m-1)$. row of $B$, then let $Z_q$ be this row.
        ELSE $Z_q = 0$.
Compute global sum $[R, Z] = \sum_{q=1}^{p} [R_q, Z_q]$ by data exchange of over all processors.
$\tilde{B} = Z - \beta R$.
FOR all $q \in \{1, \ldots, p\}$.
        $\underline{B}_q = \underline{B}_q - \frac{1}{2} \overline{v}_q R$.
        IF $\underline{B}_q$ contains the $(m-1)$. row of $B$, then set this row to 0.
$\Rightarrow \left[\tilde{B}_{m-1,0}, \ldots, \tilde{B}_{m-1,m-2}\right] = \tilde{B}$, $[B_{m,0}, \ldots, B_{m,m-2}] = B$.
```

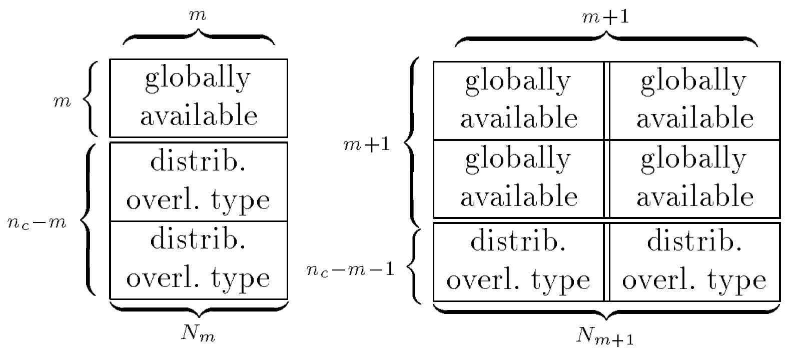
For the second part we note that in order compute $B_{m,m-1}$, $B_{m-1,m}$ and $\tilde{S}_{c,m-1}$ we have to compute $Y_{m-1} = \hat{U}_0 \cdots \hat{U}_{m-2} \tilde{U}_{m-1}$ from (8.3). The computation of Y_{m-1} no longer requires the expensive successive application of elementary Householder reflectors. If the product of Householder reflectors has been collected, then Y_{m-1} corresponds to s_{m-1} columns of Q_{m-1}. For simplicity of representation we will again assume that $s_0 = \cdots = s_m = 1$. Then Y_{m-1} will be column m of Q_{m-1}. Let $V_{m-1} = (v_{lk})_{l=1,\ldots,n_c, k=0,\ldots,m-1}$, then

$$Y_{m-1} = Q_{m-1} e_m = e_m - V_{m-1} T_{m-1}^{-1} (v_{mk})_{k=0,\ldots,m-1}^T. \tag{8.24}$$

For this operation no additional scalar product is necessary, but we should ensure that $(v_{mk})_{k=0,\ldots,m-1}$ is available on all processors. To ensure this we have to proceed for T_{m-1}, V_{m-1} analogously to B_m. Set

$$N_m = \left(\begin{array}{cccc} T_{0,0} & T_{0,1} & \cdots & T_{0,m} \\ v_{1,0} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & T_{m-1,m} \\ v_{m,0} & \cdots & v_{m,m-1} & T_{m,m} \\ \hline v_{m+1,0} & \cdots & \cdots & v_{m+1,m} \\ \vdots & & & \vdots \\ v_{n_c,0} & \cdots & \cdots & v_{n_c,m} \end{array} \right).$$

Note that by construction of Householder reflectors in Algorithm 8.10, V_m is a lower triangular matrix with unit diagonal. For N_m and N_{m+1} we will construct a similar memory distribution as for B_m, B_{m+1}:



In order to make $(v_{mk})_{k=0,\dots,m-1}$ globally available we can slightly modify Algorithm 8.15.

Algorithm 8.25 (Product of Householder Reflectors)

```
Let $V \equiv V_{m-1}, v \equiv v_m$.
FOR all $q \in \{1,\dots,p\}$.
        Compute local scalar product of $R = v^T V$:
        $\underline{v}_q := \frac{1}{2}\overline{v}_q,\ R_q = \underline{v}_q^T \overline{V}_q$.
        Get row $m$ of distributed stored matrix $V$:
        IF    $\overline{V}_q$ contains the $m$-th row of $V$, then let $Z_q$ be this row.
        ELSE $Z_q = 0$.
Compute global sum $[R, Z] = \sum_{q=1}^{p} [R_q, Z_q]$ by data exchange of over all processors.
$\Rightarrow [v_{m,0},\dots,v_{m,m-1}] = \frac{1}{2}Z,\ [T_{0,m},\dots,T_{m-1,m}] = R,\ T_{m,m} = 1/\beta_m$.
```

Finally as third part we will discuss the computation of $B_{m,m-1}$, $B_{m-1,m}$ and $\tilde{S}_{c,m-1}$ using the special distribution of $\tilde{B}_{lk}$, $\tilde{B}_{kl}$, $B_{m,k}$, $B_{k,m}$ and V_{m-1}. Recall that

$$\begin{aligned}
Y_{m-1} &= \hat{U}_0 \cdots \hat{U}_{m-2}\tilde{U}_{m-1}, \\
B_{m,m-1} &= \hat{U}_{m-1}^T \hat{U}_{m-2}^T \cdots \hat{U}_0^T S_c Y_{m-1} - \sum_{l=0}^{m-2} B_{m,l}\tilde{S}_{c,l}^{-1}\tilde{B}_{l,m-1}, \\
B_{m-1,m} &= Y_{m-1}^T S_c \hat{U}_0 \cdots \hat{U}_{m-2}\hat{U}_{m-1} - \sum_{l=0}^{m-2} \tilde{B}_{m-1,l}\tilde{S}_{c,l}^{-1} B_{l,m}, \\
\tilde{S}_{c,m-1} &= Y_{m-1}^T S_c Y_{m-1} - \sum_{l=0}^{m-2} \tilde{B}_{m-1,l}\tilde{S}_{c,l}^{-1}\tilde{B}_{l,m-1}.
\end{aligned}$$

For the computation of these values we can formulate the following algorithm.

```
**Algorithm 8.26 (New Low Rank Update)**
Let $y \equiv Y_{m-1}, v \equiv (v_{m,0},\dots,v_{m,m-1})^T, T_{m-1} \equiv T, V \equiv V_{m-1}, B \equiv B_{m,m-1}, B' \equiv B_{m-1,m}, \tilde{S}_c \equiv \tilde{S}_{c,m-1}$.

Computation of $y$:
FOR all $q \in \{1,\dots,p\}$:
        Solve $Tz = v$, $y = 0$.
        If $\overline{y}_q$ contains the $m$–th component of $y$, then set this component to 1.
        $\overline{y}_q := \overline{y}_q - \overline{V}_q z$.

Computation of $B, B', \tilde{S}_c$:
Perform $c = S_c y$, $d^T = y^T S_c$ applying Algorithm 7.52 with $\overline{X} = \overline{y}, \overline{y}^T$.

Compute $R = y^T c$ using Algorithm 7.50 with $\overline{X} = \overline{y}$, $\underline{Y} = \underline{c}$.
Multiply $c, d$ by the product of Householder reflectors applying Algorithm 8.17 with $\underline{X} = [c, d]$ and denote the result again by $[c, d]$.
The global sum in Algorithm 7.50 and Algorithm 8.17 can be carried out simultaneously.

Update $\tilde{S}_c, B, B'$:
$a = \left[\tilde{S}_{c,0}^{-1}\tilde{B}_{0,m-1},\dots,\tilde{S}_{c,m-2}^{-1}\tilde{B}_{m-2,m-1}\right]^T$, $b = \left[\tilde{B}_{m-1,0}\tilde{S}_{c,0}^{-1},\dots,\tilde{B}_{m-1,m-2}\tilde{S}_{c,m-2}^{-1}\right]^T$.
$\tilde{S}_c = R - \left[\tilde{B}_{m-1,0},\dots,\tilde{B}_{m-1,m-2}\right] a$
$B = c - [B_{m,0},\dots,B_{m,m-2}]\, a$, $B' = (d - \left[\tilde{B}_{0,m-1}^T,\dots,\tilde{B}_{m-2,m-2}^T\right] b)^T$.
Set the entry $m-1$ of $B, B'$ to 0.
```

To end up with a parallel version of the nested divide & conquer algorithm (3.15),(3.12) we obtain the following algorithm, which summarizes the previous steps.

```
**Algorithm 8.27 (Parallel nested D & C)** Consider a modified block Jacobi–splitting from (7.10). Assume that $A_{qr}, F_q^{\{q,r\}}, G_q^{\{q,r\}}, F_r^{\{q,r\}}, G_r^{\{q,r\}}$ are stored on processor $q$ and $r$ at the same time, $q, r = 1,\dots,p$.
FOR $m = 0, 1, 2, \dots$
        Consider $u \equiv \tilde{U}_m \in \mathbb{R}^{n_c} \setminus \{0\}$ and assume that the first $m$ rows of $u$ are zero and that $u$ is given by its overlapping representation.
        Step 1. Compute the Householder representation $H = I - \beta vv^T$ of $u$ applying
                Algorithm 8.10.

        Step 2. Compute column $m$ of $T_m$ and make $v_{m,0},\dots,v_{m,m}$ globally available
                using Algorithm 8.25.
        Step 3. Update $[B_{m,0},\dots,B_{m-1,m-2}]$, $\left[B_{0,m}^T,\dots,B_{m-2,m-1}^T\right]$ and make
                $\left[\tilde{B}_{m-1,0},\dots,\tilde{B}_{m-1,m-2}\right]$, $\left[\tilde{B}_{0,m-1},\dots,\tilde{B}_{m-2,m-1}^T\right]$ globally available by
                calling Algorithm 8.23.
        The global sum in Step 2 and 3 can be performed simultaneously.

        Step 4. Compute $B_{m-1,m}, B_{m,m-1}, \tilde{S}_{c,m-1}$ and make $\tilde{S}_{c,m-1}$ globally available using
                Algorithm 8.26.
```

Algorithm 8.27 provides the matrices $\tilde{B}_{kl}, \tilde{B}_{lk}$, B_{lm}, B_{ml} and $\tilde{S}_{c,l}$ for the matrix–vector multiplication with $S_{c,m}$ in Algorithm 8.20. This shows that the parallel treatment of $S_{c,m}$ can be traced back to the parallel treatment of the initial coupling system S_c by using collected products of low rank modifications. Once the collected products are used for the product of Householder reflectors and once they are used for the low rank updates of the coupling system.

8.5 Parallel Application of S_m^{-1}

As a consequence of the representation of $S_{c,m}$ in Section 4 we can easily solve a system with S_m. Again we will assume for simplicity that $s_0 = \cdots = s_m = 1$. Let $\tilde{Q}_{m-1} = Q_{m-1}[e_1, \dots, e_m]$, then

$$\begin{aligned} S_m^{-1} &= (S - F\tilde{Q}_{m-1}\tilde{Q}_{m-1}^T G)^{-1} \\ &= S^{-1} + S^{-1}F\tilde{Q}_{m-1}(\tilde{Q}_{m-1}^T S_c \tilde{Q}_{m-1})^{-1}\tilde{Q}_{m-1}^T G S^{-1}. \end{aligned} \tag{8.28}$$

But for $\tilde{Q}_{m-1}^T S_c \tilde{Q}_{m-1}$ we obtain the LU decomposition by taking the upper left $m \times m$ part of L, D, R from Lemma 3.18, which consist of the matrices $\tilde{B}_{kl}$, $\tilde{B}_{lk}$ and $\tilde{S}_{c,l}$, $0 \leqslant l < k < m$. But this part is globally available. We denote these matrices by $\tilde{L}_{m-1}, \tilde{D}_{m-1}, \tilde{R}_{m-1}$, i.e.,

$$\tilde{Q}_{m-1}^T S_c \tilde{Q}_{m-1} = \tilde{L}_{m-1}\tilde{D}_{m-1}\tilde{R}_{m-1}.$$

Since the first $m \times m$ part of V_{m-1} is also available on all processors, $\tilde{Q}_{m-1}$ can be directly accessed. We denote this upper $m \times m$ part of V_{m-1} by $\tilde{V}_{m-1}$. In this case we have

$$\tilde{Q}_{m-1} = [e_1, \dots, e_m] - V_{m-1}T_{m-1}^{-1}\tilde{V}_{m-1}^T$$

The application of $\tilde{Q}_{m-1}, \tilde{Q}_{m-1}^T$ will be done in the following two algorithms. For the multiplication with $\tilde{Q}_{m-1}$ we will end up with an overlapping type vector, while for $\tilde{Q}_{m-1}^T$ we assume that the right hand side is given by an adding type vector.

Algorithm 8.29 (Multiplication with $\tilde{Q}_{m-1}$)

```
Let $\tilde{V} \equiv \tilde{V}_{m-1}, V \equiv V_{m-1}, T \equiv T_{m-1}$ and compute $X = \tilde{Q}_{m-1}Z$.
FOR $l = 0, \dots, m-1$
    IF    $\overline{X}_q$ contains the $(l+1)$. component of $X$ then set this component to
          the $(l+1)$. component of $Z$.
    ELSE  Set component $l+1$ of $\overline{X}_q$ to 0.
FOR all $q \in \{1, \dots, p\}$
    $\overline{X}^{(q,r)} = \overline{X}^{(q,r)} - \overline{V}^{(q,r)}(T^{-1}(\tilde{V}^T Z))$ for any $r$ such that $\{q,r\} \in \mathfrak{I}$.
$\Longrightarrow X$ is given by its overlapping representation.
```

```
**Algorithm 8.30 (Multiplication with $\tilde{Q}_{m-1}^T$)**
Let $\tilde{V} \equiv \tilde{V}_{m-1}, V \equiv V_{m-1}, T \equiv T_{m-1}$ and compute $Z = \tilde{Q}_{m-1}^T X$.
FOR $l = 0, \dots, m-1$
        IF    $\underline{X}_q$ contains the $(l+1)$. component of $X$ then let $Z_{lq}$ be this component.
        ELSE Set component $l+1$ of $\underline{X}_q$ to 0.
FOR all $q \in \{1, \dots, p\}$
        $R_q = \overline{V}_q^T \underline{X}_q$
Compute $[R, Z] = \sum_{q=1}^{p} [R_q, Z_{0,q}, \dots, Z_{m-1,q}]$ by exchange of over all processors.
$Z = Z - \tilde{V} T^{-T} R$.
```

This simplifies the application of S_m^{-1}, since only the application of $\tilde{Q}_{m-1}^T$ requires one step of global communication.

We assume that $b = (b_q)_{q=1,\dots,p} \in \mathbb{R}^n$ is distributed such that b_q lies on processor q. The solution of $S_m^{-1} b$ will be denoted by $x = (x_q)_{q=1,\dots,p}$.

```
**Algorithm 8.31 (Solving a system $S_m x = b$ in parallel)**
Compute $X = G S^{-1} b$:
FOR all $q \in \{1, \dots, p\}$
        $\underline{X}^{(q,r)} = G_q^{\{q,r\}} S_{qq}^{-1} b_q$, for any $r$ such that $\{q,r\} \in \mathbf{i}$.
$\Longrightarrow$ $X$ is given by an adding representation.
Compute $Z = \tilde{Q}_{m-1}^T X$ applying Algorithm 8.30.
Solve $\tilde{L}_{m-1} \tilde{D}_{m-1} \tilde{R}_{m-1} e = Z$.
Compute $X = \tilde{Q}_{m-1}^T e$ applying Algorithm 8.29 with $Z := e$ and result $\overline{X}$.
Final update $x = S^{-1}(b + FX)$:
FOR all $q \in \{1, \dots, p\}$
        $x_q = S_{qq}^{-1} \left( b_q + \sum_{r:\, \{q,r\} \in \mathfrak{I}} F_q^{\{q,r\}} \overline{X}^{(q,r)} \right)$.
```

In contrast to Algorithm 3.15,3.12 we do not need any more the matrix $E_k = S_k^{-1} \tilde{F}_k$ from (8.1). Instead we have introduced the matrices $\tilde{B}_{kl}, \tilde{B}_{lk}, B_{lm}, B_{ml}$ which have several advantages in the parallel realization of the nested divide & conquer method. In addition B_{lm}, B_{ml} typically need much less storage than E_k, since they are only of the same order as the coupling system.

So far we have not discussed the choice of $\tilde{U}_k$. For the positive definite case we have by Lemma 3.31, that skillful linear combinations of eigenvectors are optimal with respect to the condition number of the remaining system. We can compute approximate eigenvectors using Lanczos' method[67]. For the unsymmetric case it is still open, which orthogonal transformation should be used. On one hand an orthogonal transformation which corresponds to an invariant subspace may be useful in combination with a preconditioner, but on the other hand this may be hard to obtain. Another way can be to take $\tilde{U}_k$ from the Hermitian part or skew–Hermitian of $S_{c,k}$ in order to modify the field of values.
If $\tilde{U}_k$ is taken from the Arnoldi process, then the computation of $B_{m,m-1}$ in (8.22) can be

simplified since $B_{m,m-1} = \hat{U}^T_{m-1} S_{c,m-1} \tilde{U}_{m-1}$. The Arnoldi process [41],pp.501–502, generates a sequence of the following form.

$$S_{c,m-1} V_t = V_{t+1} \hat{H}_t$$

where $V_t = [v_1, \dots, v_t]\,, V_{t+1} = [V_t, v_{t+1}]$ and V_{t+1} satisfies $V^T_{t+1} V_{t+1} = I$.

$$\hat{H}_t = \left[\begin{array}{ccc} & H_t & \\ \hline 0 \ \cdots \ 0 & & h_{t+1,t} \end{array} \right],$$

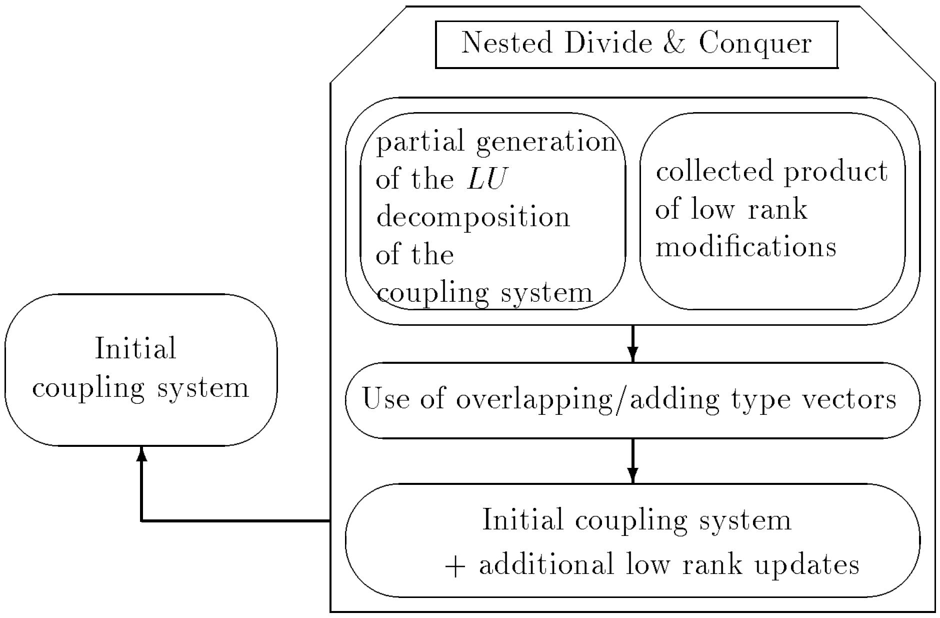
where H_t is an upper Hessenberg matrix. In this case $\tilde{U}_{m-1}$ will have the form $\tilde{U}_{m-1} = V_t \tilde{V}$. From this it follows that

$$B_{m,m-1} = \hat{U}^T_{m-1} S_{c,m-1} V_t \tilde{V} = \hat{U}^T_{m-1} (V_{t+1} \hat{H}_t \tilde{V}).$$

But $V_{t+1} \hat{H}_t \tilde{V}$ is typically much easier to compute that $S_{c,m-1} \tilde{U}_{m-1}$ which shows that the computation of $B_{m,m-1}$ can be essentially simplified.
Unless $S_{c,m-1}$ is symmetric, $B_{m-1,m}$ still has to be computed using (8.22). In the symmetric case only $B_{l,m-1}$ with $l \geqslant m$ is necessary for symmetry reasons.

Summary

The parallel realization of nested divide & conquer methods has been discussed in detail. For the update matrix $\tilde{U}_k$ its Householder representation has been used. Since the successive use of Householder reflectors involves several steps of communication, the reflectors have been collected in a single matrix reducing the data traffic to only one communication step each time when they are applied. The same collected representation has been made for S_k^{-1} and $S_{c,k}$. To get this, additional matrices have been introduced which can be seen as part of the nested LU–decomposition of S_c. Their generation and the way one has to work with these matrices has been described in Algorithm 8.27. The algorithm has taken care of reducing communication time by collecting operations which can be done simultaneously and performing the communication afterwards. The introduction of the additional matrices has turned out to simplify the use of nested divide & conquer methods in parallel computations.
The parallel realization of nested divide & conquer methods can be summarized in the following table.


Nested Divide & Conquer
partial generation of the LU decomposition of the coupling system
collected product of low rank modifications
Use of overlapping/adding type vectors
Initial coupling system + additional low rank updates
Initial coupling system

Chapter 9

Numerical Results

In this chapter we will illustrate the theory that has been presented for several numerical examples.

We will start with the symmetric positive definite case. In this case we will first examine the modified block Jacobi splittings from Chapter 6. Then we will present numerical results for the corresponding parallel realization.

As third part we will discuss some unsymmetric examples. For these examples we will also examine the modified block Jacobi splittings from Chapter 6. In addition we will illustrate how the nested Divide & Conquer strategy can be applied in these examples.

The programs that have been used are on a floppy disk, which has been added to this paper.

9.1 Numerical Examples on Modified Block Jacobi Splittings

To show the improvements of the coupling system S_c when using modified block Jacobi splittings from Chapter 6 we will examine several examples. We will compare the unmodified block Jacobi splitting from Definition 5.2 with the modified block Jacobi splitting constructed by Algorithm 6.89 and the modified splitting obtained by Algorithm 6.90, that is the approach which does not require the parameter α from (6.49).
The case when C from Lemma 6.19 is singular will be illustrated for an example.

We denote by $\kappa_2(B) = \lambda_{\max}(B)/\lambda_{\min}(B)$ the condition number of B. For $S_J^{-1}A$ this will be the condition number of $S_J^{-1/2}AS_J^{-1/2}$, where S_J is the block diagonal part of A. $S_{c,opt}$ will be the coupling system with respect to the optimal block diagonal modification from Algorithm 6.89. S_{c,f_o} will be the coupling system from Algorithm 6.90.
In all these examples we will compare the block Jacobi method with Algorithm 6.89 and Algorithm 6.90 for various numbers p of blocks, $p = 2, 4, 8, 16, \ldots$. The relation between p and the size n of the system together with the size of the blocks will restrict the freedom in

choosing p. In many examples the maximum number p of blocks will be $\leqslant 8$. Therefore the requirement to have a small coupling system compared with the size of the initial system is not fulfilled and the computation time in calculating the modified block diagonal splitting will be expensive. However these examples can be used to illustrate the improvement of the coupling system.

The computation were carried out using **MATLAB** [60]. For the computation of the parameter α in Algorithm 6.89 we used **MATLAB** 's 'eig' function.
Note that by computing $\bar{D}, D$ from Lemma 6.19 and X we get an explicit representation of S_c in (6.9). This will reduce the number of flops when applying a matrix–vector multiplication $S_c \cdot x$, e.g. using the cg–method.

For any example we will compare the following four topics.

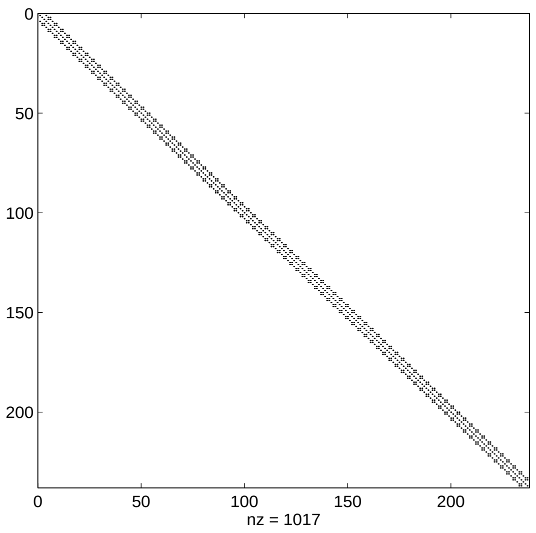
1. the condition number κ_2 of $S_J^{-1}A$ with those of $S_{c,opt}, S_{c,fo}$.
2. the number of iterations needed by the CG–method
3. the number of sequential floating point operations(flops) for the LU–decomposition versus the flops required for the generation of X in Algorithm 6.90.
4. the number of flops for the iterative solution process for the block Jacobi method versus the number of flops required by Algorithm 6.90.

As stopping criterion for the solution process we will use $\|r_k\| \leqslant \sqrt{\text{eps}}\|r_0\|$, where r_k is the residual in step k. Here eps $\approx 2.2204 \cdot 10^{-16}$. The iterative solution is performed ten times for random right hand sides and finally the average is taken. As initial guess we will choose $x_0 = 0$.

We will consider several examples from the Harwell–Boeing sparse matrix collection [28]. The test matrices can be accessed via anonymous ftp from **ftp.orion.cerfacs.fr**.

Example 9.1

The matrix **LANPRO/NOS1** *is symmetric positive definite, its size is $n = 237$ and it is block tridiagonal with all blocks of size 3×3. Its pattern is illustrated in the picture on the right hand side. We will apply the three methods to this this matrix for various number of blocks $p = 2, 4, 8, 16, 32$.*



The condition number of the preconditioned matrix is given in the following table.

Condition Number

$\kappa_2 \setminus p$	*2*	*4*	*8*	*16*	*32*
$\kappa_2(S_J^{-1}A)$	$1.4{\cdot}10^5$	$3.4{\cdot}10^6$	$6.4{\cdot}10^6$	$1.2{\cdot}10^7$	$2.2{\cdot}10^7$
$\kappa_2(S_{c,opt})$	$1.3{\cdot}10^0$	$4.3{\cdot}10^1$	$7.4{\cdot}10^2$	$1.3{\cdot}10^4$	$4.2{\cdot}10^5$
$\kappa_2(S_{c,fo})$	$4.0{\cdot}10^0$	$6.2{\cdot}10^1$	$1.1{\cdot}10^3$	$1.8{\cdot}10^4$	$3.4{\cdot}10^5$

From the comparison of the condition numbers we expect a remarkable difference in the number of iterations for the CG–method, at least for smaller numbers p.

Number of Iteration Steps

it. $\setminus p$	*2*	*4*	*8*	*16*	*32*
$S_J^{-1}A$	*9*	*29*	*80*	*171*	*324*
$S_{c,opt}$	*4*	*10*	*24*	*53*	*200*
$S_{c,fo}$	*4*	*10*	*24*	*56*	*188*

In spite of a large condition number the number of iterations is relatively small. This is probably related to the low rank property here. For $S_J^{-1}A$ *the rank rank of the remaining matrix will be* $6(p-1)$ *and the size of* S_c *will be* $3(p-1)$*. Again the condition number and the number of iterations for Algorithm 6.89,Algorithm 6.90 are quite close to each other. At last we will compare the number of* flops *. We will split this into two parts. The first part will be the number of* flops *for the Cholesky decomposition compared with the number of* flops *for the generation of X from Algorithm 6.90. The second part will be the number of* flops *for the solution process.*

flops *for Cholesky Decomposition Versus the Generation of X*

flops $\setminus p$	*2*	*4*	*8*	*16*	*32*
Cholesky decomposition S_J	$3.8{\cdot}10^3$	$3.6{\cdot}10^3$	$3.4{\cdot}10^3$	$2.2{\cdot}10^3$	$1.7{\cdot}10^3$
Alg. 6.90 $S_{c,fo}$	$7.8{\cdot}10^3$	$1.2{\cdot}10^4$	$2.5{\cdot}10^4$	$6.9{\cdot}10^4$	$2.5{\cdot}10^5$

flops *for the Solution Process*

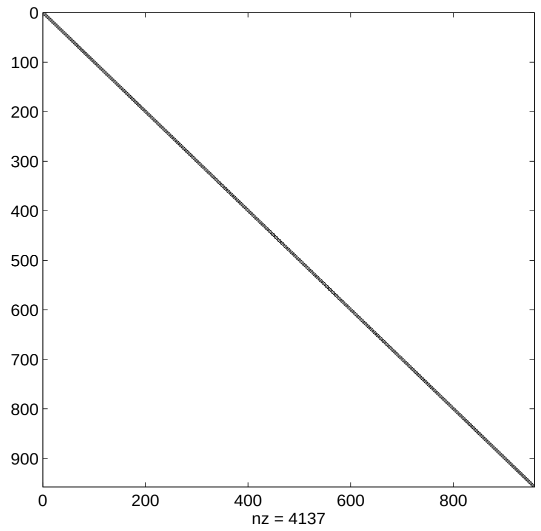
flops $\setminus p$	*2*	*4*	*8*	*16*	*32*
$S_J^{-1}A$	$7.4{\cdot}10^4$	$2.4{\cdot}10^5$	$6.4{\cdot}10^5$	$1.3{\cdot}10^6$	$2.3{\cdot}10^6$
$S_{c,fo}$	$1.2{\cdot}10^4$	$1.5{\cdot}10^4$	$3.4{\cdot}10^4$	$1.3{\cdot}10^5$	$9.2{\cdot}10^5$

Total Amount in flops

flops $\setminus p$	*2*	*4*	*8*	*16*	*32*
$S_J^{-1}A$	$7.8{\cdot}10^4$	$2.4{\cdot}10^5$	$6.5{\cdot}10^5$	$1.3{\cdot}10^6$	$2.3{\cdot}10^6$
$S_{c,fo}$	$2.0{\cdot}10^4$	$2.7{\cdot}10^4$	$6.0{\cdot}10^4$	$2.0{\cdot}10^5$	$1.2{\cdot}10^6$

As expected, the generation of X will be more expensive than the pure Cholesky decomposition. But the additional number of flops *is moderate since only 6 right hand sides have to be solved with S_J in Algorithm 6.90. As side effect of the generation of X, S_c is explicitly computed and thus a step of the solution process for S_c will need much less* flops *than the corresponding solution process for the block Jacobi method. This compensates the more expensive generation of X. In addition the number of iterations has been significantly reduced when the CG method is applied to $S_{c,fo}$.*

Example 9.2
The matrix **LANPRO/NOS2** *is symmetric positive definite, its size is $n = 957$ and it is block tridiagonal with all blocks of size 3×3. We will examine this matrix for various number of blocks $p = 2, 4, 8, 16, 32, 64$.*



The condition number of the preconditioned matrix is given in the following table.

Condition Number

$\kappa_2 \backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$\kappa_2(S_J^{-1}A)$	$8.4{\cdot}10^6$	$8.1{\cdot}10^8$	$1.6{\cdot}10^9$	$3.1{\cdot}10^9$	$5.9{\cdot}10^9$	$1.1{\cdot}10^{10}$
$\kappa_2(S_{c,opt})$	$1.3{\cdot}10^0$	$4.1{\cdot}10^1$	$6.4{\cdot}10^2$	$1.0{\cdot}10^4$	$1.7{\cdot}10^5$	$3.3{\cdot}10^6$
$\kappa_2(S_{c,fo})$	$4.0{\cdot}10^0$	$7.2{\cdot}10^1$	$1.2{\cdot}10^3$	$2.0{\cdot}10^4$	$3.2{\cdot}10^5$	$2.8{\cdot}10^6$

Like in Example 9.1 the condition number is clearly improved by Algorithm 6.89 and Algorithm 6.90. Next we compare the number of iteration steps in the cg–method.

Number of Iteration Steps

p	*2*	*4*	*8*	*16*	*32*	*64*
$S_J^{-1}A$	*11*	*38*	*111*	*359*	*1018*	*2271*
$S_{c,opt}$	*4*	*10*	*24*	*54*	*161*	*527*
$S_{c,fo}$	*4*	*10*	*25*	*63*	*194*	*501*

Due to the low rank property the big condition number does not affect the problem for small p. But for larger p the number of iterations increases drastically. For Algorithm 6.89,6.90 *this effect is not so critical as for the block Jacobi method. But for larger numbers of p the improvement is weaker.*

Again we will compare the number of flops *for the Cholesky decomposition versus the generation of X and for the iterative solution process.*

flops *for Cholesky Decomposition Versus the Generation of X*

flops $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
Cholesky decomposition S_J	$1.6{\cdot}10^4$	$1.6{\cdot}10^4$	$1.6{\cdot}10^4$	$1.5{\cdot}10^4$	$1.4{\cdot}10^4$	$9.1{\cdot}10^3$
Alg. 6.90 $S_{c,fo}$	$3.0{\cdot}10^4$	$3.9{\cdot}10^4$	$5.7{\cdot}10^4$	$1.0{\cdot}10^5$	$2.8{\cdot}10^5$	$9.7{\cdot}10^5$

flops *for the Solution Process*

flops $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$S_J^{-1}A$	$3.7{\cdot}10^5$	$1.3{\cdot}10^6$	$3.8{\cdot}10^6$	$1.2{\cdot}10^7$	$3.3{\cdot}10^7$	$6.8{\cdot}10^7$
$S_{c,fo}$	$4.7{\cdot}10^4$	$5.1{\cdot}10^4$	$7.1{\cdot}10^4$	$1.8{\cdot}10^5$	$9.9{\cdot}10^5$	$5.9{\cdot}10^6$

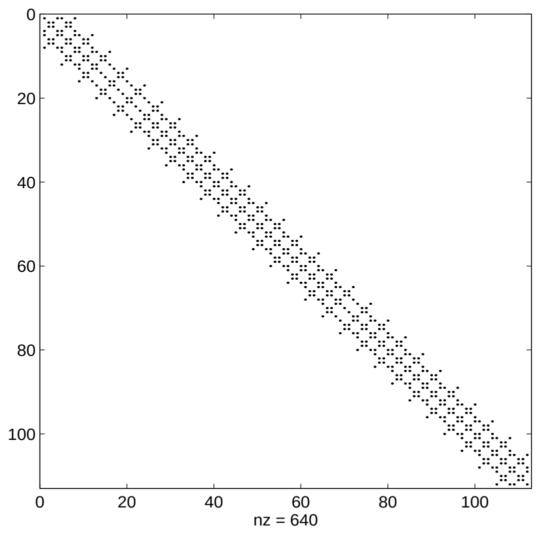
Total Amount in flops

flops $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$S_J^{-1}A$	$3.9{\cdot}10^5$	$1.3{\cdot}10^6$	$3.8{\cdot}10^6$	$1.2{\cdot}10^7$	$3.3{\cdot}10^7$	$6.8{\cdot}10^7$
$S_{c,fo}$	$7.7{\cdot}10^4$	$9.0{\cdot}10^4$	$1.3{\cdot}10^5$	$2.8{\cdot}10^5$	$1.3{\cdot}10^6$	$6.9{\cdot}10^6$

Algorithm 6.90 needs significantly less flops *than the block Jacobi method. The fact that the system is very ill–conditioned extremely affects the block Jacobi method, while the influence of the huge condition number is much less for Algorithm 6.89,6.90. Like in Example 9.1 the additional amount for generating X is quite moderate, since for the computation of $\bar{D}, D$ 6 right hand sides have to be solved. This is neglectible compared with the number of iterations for the solution process.*

Example 9.3

The matrix **BCSSTRUC1/BCSSTK03** *is symmetric positive definite, its size is* $n = 112$ *and it is block tridiagonal with all blocks of size* 4. *Its pattern is illustrated in the picture on the right hand side. We will examine this matrix for various number of blocks* $p = 2, 4, 8$.



The condition number of the preconditioned matrix is given in the following table.

Condition Number

$\kappa_2 \setminus p$	*2*	*4*	*8*
$\kappa_2(S_J^{-1}A)$	$1.8{\cdot}10^4$	$5.2{\cdot}10^4$	$2.0{\cdot}10^5$
$\kappa_2(S_{c,opt})$	$1.3{\cdot}10^0$	$1.8{\cdot}10^0$	$2.9{\cdot}10^1$
$\kappa_2(S_{c,fo})$	$1.2{\cdot}10^0$	$1.8{\cdot}10^0$	$2.3{\cdot}10^1$

The condition number of the coupling systems $S_{c,opt}, S_{c,fo}$ is extremely improved compared with the condition number of $S_J^{-1}A$. Thus we expect a remarkable difference in the number of iteration for the CG–method.

Number of Iteration Steps

it. $\setminus p$	*2*	*4*	*8*
$S_J^{-1}A$	*8*	*22*	*53*
$S_{c,opt}$	*4*	*9*	*23*
$S_{c,fo}$	*4*	*9*	*20*

Although the condition number has been significantly improved by Algorithm 6.89,6.90, the number of iterations of the block Jacobi method is moderate. Here again the low rank will reduce the number iteration steps. At last we will compare the number of flops .

flops *for Cholesky Decomposition Versus the Generation of X*

flops $\setminus p$		*2*	*4*	*8*
Cholesky decomposition	S_J	$2.1{\cdot}10^3$	$1.8{\cdot}10^3$	$1.2{\cdot}10^3$
Alg. 6.90	$S_{c,fo}$	$5.7{\cdot}10^3$	$1.2{\cdot}10^4$	$3.3{\cdot}10^4$

flops *for the Solution Process*

flops $\setminus p$	*2*	*4*	*8*
$S_J^{-1}A$	$3.6{\cdot}10^4$	$9.4{\cdot}10^4$	$2.1{\cdot}10^5$
$S_{c,fo}$	$6.8{\cdot}10^3$	$1.1{\cdot}10^4$	$3.0{\cdot}10^4$

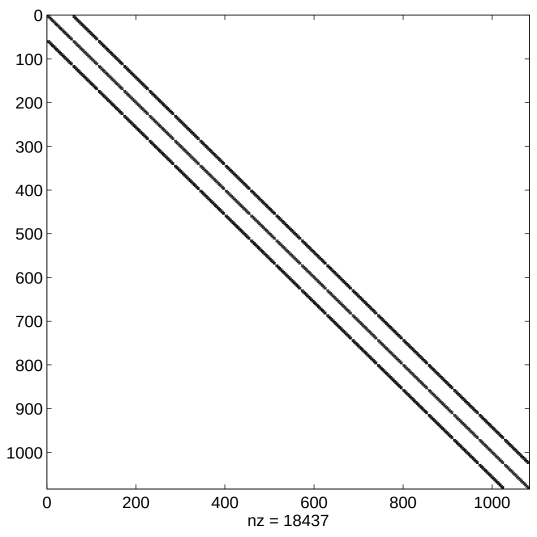
Total amount in flops

flops $\setminus p$	*2*	*4*	*8*
$S_J^{-1}A$	$3.8{\cdot}10^4$	$9.5{\cdot}10^4$	$2.1{\cdot}10^5$
$S_{c,fo}$	$1.2{\cdot}10^4$	$2.3{\cdot}10^4$	$6.3{\cdot}10^4$

The more expensive generation of X in Algorithm 6.90 compared with the Cholesky decomposition for the block Jacobi method will be equalized by the iterative solution process. The overhead for generating X is moderate (8 right hand sides) while the number of iterations will be clearly reduced.

Example 9.4

The matrix **BCSSTRUC1/BCSSTK09** *is symmetric positive definite, its size is $n = 1083$ and it is block tridiagonal with blocks of size 57×57. Its pattern is illustrated in the picture on the right hand side. We will examine this matrix for various number of blocks $p = 2, 4, 8$.*



The condition number of the preconditioned matrix is given in the following table.

Condition Number

$\kappa_2 \setminus p$	*2*	*4*	*8*
$\kappa_2(S_J^{-1}A)$	$3.2{\cdot}10^3$	$6.6{\cdot}10^3$	$1.3{\cdot}10^4$
$\kappa_2(S_{c,opt})$	$1.3{\cdot}10^0$	$3.8{\cdot}10^1$	$1.3{\cdot}10^3$
$\kappa_2(S_{c,fo})$	$1.1{\cdot}10^0$	$3.1{\cdot}10^1$	$1.1{\cdot}10^3$

From the comparison of the condition numbers we expect a clear improvement in the number of iteration for the CG–method at least for $p = 2$ and $p = 4$. In fact this will be the case as the following table will show.

Number of Iteration Steps

it. $\setminus p$	*2*	*4*	*8*
$S_J^{-1}A$	*50*	*113*	*172*
$S_{c,opt}$	*6*	*22*	*118*
$S_{c,fo}$	*5*	*21*	*102*

Again we will compare the number of flops .

flops *for Cholesky Decomposition Versus the Generation of X*

flops $\setminus p$		*2*	*4*	*8*
Cholesky decomposition	S_J	$1.7{\cdot}10^6$	$7.1{\cdot}10^5$	$1.8{\cdot}10^5$
Alg. 6.90	$S_{c,fo}$	$9.1{\cdot}10^6$	$2.5{\cdot}10^7$	$7.7{\cdot}10^7$

flops *for the Solution Process*

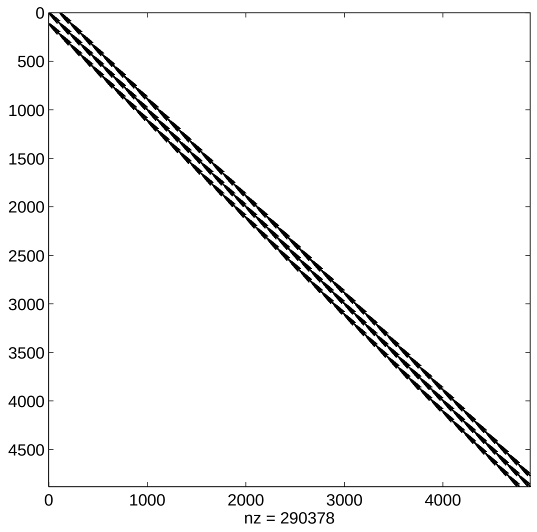
flops $\backslash p$	*2*	*4*	*8*
$S_J^{-1}A$	$1.0{\cdot}10^7$	$1.7{\cdot}10^7$	$1.8{\cdot}10^7$
$S_{c,fo}$	$5.7{\cdot}10^5$	$2.1{\cdot}10^6$	$2.1{\cdot}10^7$

Total amount in flops

flops $\backslash p$	*2*	*4*	*8*
$S_J^{-1}A$	$1.2{\cdot}10^7$	$1.8{\cdot}10^7$	$1.8{\cdot}10^7$
$S_{c,fo}$	$9.7{\cdot}10^6$	$2.7{\cdot}10^7$	$9.8{\cdot}10^7$

The fact that the size of the off–diagonal blocks is 57 means that Algorithm 6.90 has to spend a large amount of computational cost into the generation of $D, \bar{D}$ and $\bar{D}^{-1}$ (114 right hand sides). On the other hand the size of S_c will be 396 *for $p = 8$. Compared with the size $n =$ 1083 of the initial system S_c cannot be called small. In addition the improvement in the condition number is for $p = 8$ not so great. So the number of iterations for the CG–method is not essentially reduced for $p = 8$. Here the block Jacobi method will need less* flops *than Algorithm 6.90.*

Example 9.5
The matrix **BCSSTRUC2/BCSSTK16** *is symmetric positive definite, its size is $n =$ 4884 and it is block tridiagonal with blocks of size $\leqslant$ 138 $\times$ 138. Its pattern is illustrated in the picture on the right hand side. We will examine this matrix for various number of blocks $p = 2, 4, 8, 16$.*



The condition number of the preconditioned matrix is given in the following table.

Condition Number

$\kappa_2 \backslash p$	*2*	*4*	*8*	*16*
$\kappa_2(S_J^{-1}A)$	$1.4{\cdot}10^2$	$2.3{\cdot}10^2$	$4.6{\cdot}10^2$	$9.9{\cdot}10^2$
$\kappa_2(S_{c,opt})$	$1.3{\cdot}10^0$	$1.8{\cdot}10^0$	$6.6{\cdot}10^0$	$4.6{\cdot}10^1$
$\kappa_2(S_{c,fo})$	$1.0{\cdot}10^0$	$1.8{\cdot}10^0$	$6.6{\cdot}10^0$	$4.8{\cdot}10^1$

The small condition numbers of $S_{c,opt}, S_{c,fo}$ ensure a small steps of iteration for the CG–method. We may expect an improvement in the number of iterations which is partially confirmed by the numerical observations.

Number of Iteration Steps

it. \ p	*2*	*4*	*8*	*16*
$S_J^{-1}A$	*19*	*26*	*34*	*47*
$S_{c,opt}$	*7*	*9*	*21*	*53*
$S_{c,fo}$	*5*	*9*	*20*	*54*

Although the system obtained by Algorithm 6.90,6.89 is better conditioned than the initial matrix, preconditioned by the block diagonal matrix S_J, the number of cg–iterations for this preconditioned system is small. Here we cannot expect an improvement neither in the number of iterations nor in the total amount of computational work.

flops *for Cholesky Decomposition Versus the Generation of X*

flops \ p	*2*	*4*	*8*	*16*
Cholesky decomposition S_J	$2.3{\cdot}10^8$	$1.4{\cdot}10^8$	$4.6{\cdot}10^7$	$1.2{\cdot}10^7$
Alg. 6.90 $S_{c,fo}$	$3.8{\cdot}10^8$	$4.6{\cdot}10^8$	$8.7{\cdot}10^8$	$2.5{\cdot}10^9$

flops *for the Solution Process*

flops \ p	*2*	*4*	*8*	*16*
$S_J^{-1}A$	$7.7{\cdot}10^7$	$8.8{\cdot}10^7$	$7.8{\cdot}10^7$	$7.2{\cdot}10^7$
$S_{c,fo}$	$9.6{\cdot}10^6$	$1.1{\cdot}10^7$	$2.3{\cdot}10^7$	$1.1{\cdot}10^8$

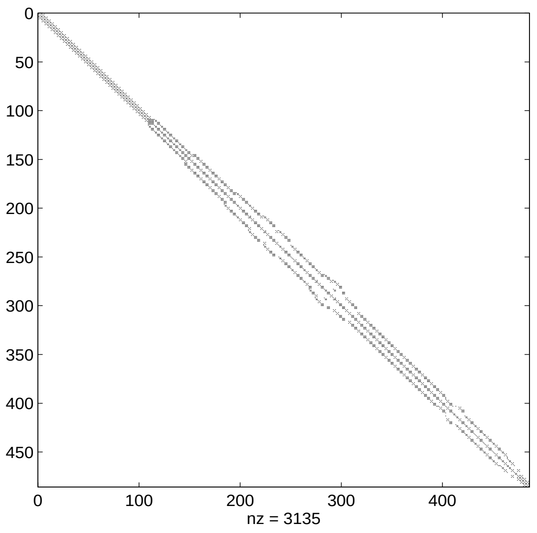
Total amount in flops

flops \ p	*2*	*4*	*8*	*16*
$S_J^{-1}A$	$3.1{\cdot}10^8$	$2.2{\cdot}10^8$	$1.2{\cdot}10^8$	$8.4{\cdot}10^7$
$S_{c,fo}$	$3.9{\cdot}10^8$	$4.7{\cdot}10^8$	$9.0{\cdot}10^8$	$2.6{\cdot}10^9$

Algorithm 6.90 cannot compete in this case, although the condition number is approximately 10^2 *less than for the block Jacobi method. The condition number itself only gives few information about the eigenvalue distribution of the corresponding matrix. But the eigenvalue distribution will be more important for the solution process than the condition number. This may be the reason for the few steps of iteration for the block Jacobi method.*

Example 9.6

The matrix **BCSSTRUC3/BCSSTK20** *is symmetric positive definite, its size is* $n = 485$ *and it is block tridiagonal with blocks of size* $\leqslant 15\times15$. *Its pattern is illustrated in the picture on the right hand side. We will examine this matrix for various number of blocks* $p = 2, 4, 8, 16$.



The condition number of the preconditioned matrix is given in the following table.

Condition Number

$\kappa_2 \backslash p$	*2*	*4*	*8*	*16*
$\kappa_2(S_J^{-1}A)$	$1.8{\cdot}10^8$	$9.4{\cdot}10^8$	$5.3{\cdot}10^{10}$	$1.2{\cdot}10^{11}$
$\kappa_2(S_{c,opt})$	$1.3{\cdot}10^0$	$1.8{\cdot}10^2$	$6.0{\cdot}10^3$	$3.1{\cdot}10^5$
$\kappa_2(S_{c,fo})$	$1.0{\cdot}10^0$	$1.2{\cdot}10^2$	$4.6{\cdot}10^3$	$1.2{\cdot}10^5$

Algorithm 6.90 and Algorithm 6.89 show a drastic improvement for the condition number of $S_{c,opt}, S_{c,fo}$ *compared with* $S_J^{-1}A$. *From this we expect that the number of iterations for the CG-method is significantly reduced by Algorithm 6.90 and Algorithm 6.89.*

Number of Iteration Steps

it. $\backslash p$	*2*	*4*	*8*	*16*
$S_J^{-1}A$	*44*	*122*	*390*	*997*
$S_{c,opt}$	*6*	*27*	*63*	*266*
$S_{c,fo}$	*5*	*25*	*63*	*291*

As expected the number of iterations for $S_{c,opt}, S_{c,fo}$ *is much less than for the block Jacobi method. At last we will compare the number of* flops .

flops *for Cholesky Decomposition Versus the Generation of X*

flops $\backslash p$	*2*	*4*	*8*	*16*
Cholesky decomposition S_J	$1.5{\cdot}10^4$	$1.2{\cdot}10^4$	$9.2{\cdot}10^3$	$7.5{\cdot}10^3$
Alg. 6.90 $S_{c,fo}$	$9.2{\cdot}10^4$	$1.3{\cdot}10^5$	$3.0{\cdot}10^5$	$9.0{\cdot}10^5$

flops *for the Solution Process*

flops $\backslash p$	*2*	*4*	*8*	*16*
$S_J^{-1}A$	$9.4{\cdot}10^5$	$2.5{\cdot}10^6$	$7.7{\cdot}10^6$	$1.9{\cdot}10^7$
$S_{c,fo}$	$3.8{\cdot}10^4$	$9.3{\cdot}10^4$	$2.8{\cdot}10^5$	$2.0{\cdot}10^6$

Total amount in flops

flops $\backslash p$	*2*	*4*	*8*	*16*
$S_J^{-1}A$	$9.6{\cdot}10^5$	$2.5{\cdot}10^6$	$7.7{\cdot}10^6$	$1.9{\cdot}10^7$
$S_{c,fo}$	$1.3{\cdot}10^5$	$2.3{\cdot}10^5$	$5.7{\cdot}10^5$	$2.9{\cdot}10^6$

The generation of X in Algorithm 6.90 requires the additional solution of up to 30 right hand sides with S_J in order to compute $\bar{D}, D$ and the size of S_c will be approximately $15, 45, 105, 225$ *for $p = 2, 4, 8, 16$ which is at least small for $p \leqslant 8$ compared with the total size $n = 485$ of the system. But on the other hand this additional amount of work will be equalized by the iterative solution process. In addition the system preconditioned by the block-Jacobi method is still ill–conditioned while the improvement for S_c ends up in much smaller number of iterations.*

Example 9.7

As final example for the use of modified block Jacobi splittings we discuss the case when the matrix C from Lemma 6.19 in Chapter 6 is singular. Throughout Chapter 6 we considered only the case when C is nonsingular. The nonsingularity of C has played an important role in deriving explicit solution for the Riccati equation (6.12). Here we will show for an example that the singular case is more technical but in principle it can also be treated similarly to the nonsingular case.

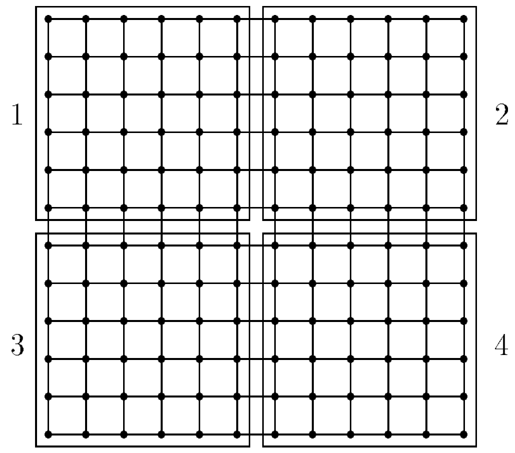
Consider the matrix

$$A_n = \begin{pmatrix} T & -I & & \\ -I & T & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & T \end{pmatrix} \in \mathbb{R}^{n,n}, \text{ where } T = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{N,N}, n = N^2.$$

This matrix has already been discussed in Example 6.3, 2.15. A arises from the discretization of the problem

$$\begin{aligned} -\Delta u &= f \text{ in } [0,1]^2 \\ u &= g \text{ on } \partial[0,1]^2 \end{aligned}$$

using five point star difference discretization [48]. Its graph can read as a checker board having the crossings as nodes. Assume that its graph is partitioned into 4 parts checker board–wise.



Each part should correspond to a diagonal block of A after a permutation P^TAP with a suitable permutation matrix P. This case is of special interest, since assumption (6.32) of Lemma 6.31 is violated and C will be singular in this case. By taking the blocks in the order $1,4,2,3$ we obtain the following system. Set $m = n/4, K = N/2$.

$$A_n \mapsto \left(\begin{array}{cc|cc} A_m & O & -E & -F \\ O & A_m & -F^T & -E^T \\ \hline -E^T & -F & A_m & O \\ -F^T & -E & O & A_m \end{array}\right),$$

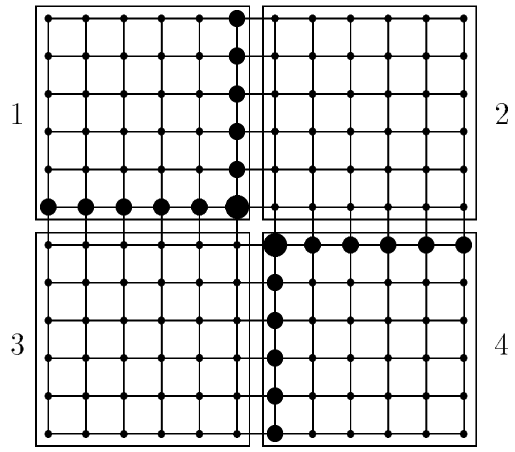
E, F are defined by

$$E = I_K \otimes (e_K e_1^T), F = (e_K e_1^T) \otimes I_K.$$

Here $\otimes$ denotes the Kronecker product (see e.g. [3], p.149). Here L from (5.12) will correspond to the nonzero columns of

$$\left(\begin{array}{cccc} E & F & & \\ & & F^T & E^T \\ \hline O & \cdots & \cdots & O \\ O & \cdots & \cdots & O \end{array}\right).$$

Denote the set of nonzero columns of this matrix by $i_1, \ldots, i_s$. The indices are illustrated by big bullets in the picture below.



The indices $i_1, \dots, i_s$ can be grouped together into four sets corresponding to the four pairs $\{1,2\}, \{2,3\}, \{4,2\}, \{4,3\}$ which will form the set $\mathfrak{I}$ from (5.5) for the block partitioning of the coupling system. These are the following entries. For the connection between subdomain 1 and 2 we have

$$K, N+K, 2N+K, \dots, (K-1)N+K$$

for the connection between subdomain 1 and 3 we have

$$(K-1)N+1, (K-1)N+2, (K-1)N+3, \dots, (K-1)N+K$$

for the connection between subdomain 3 and 4 we have

$$KN+K+1, (K+1)N+K+1, (K+2)N+K+1, \dots, (N-1)N+K+1$$

and finally we have the entries

$$KN+K+1, KN+K+2, KN+K+3, \dots, (K+1)N$$

for the connection between subdomain 2 and 4.
Here the indices with label $(K-1)N+K$ and $KN+K+1$ appear twice. From this it follows that L is rank deficient. The rank will be $2(N-1)$ instead of $2N$, but it is easy to calculate a full rank decomposition of L. For this we only have to annihilate the duplicate entries.

$$\begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \sqrt{2} & 0 \end{pmatrix}.$$

We obtain a decomposition $L = [L_1, \mathrm{O}] \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}$ where U_2 has size $N \times 2$. Up to additional zeros, U_2 will correspond to

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

The theory of Chapter 6 is based on the splitting $A = S_J - LM - M^*L^*$. By setting $M_1 = U_1^*M$ we obtain $A = S_J - L_1M_1 - M_1^*L_1^*$ with a full rank matrix L_1! So in principle we can apply the theory from Chapter 6 for the problem, where L, M are replaced by L_1, M_1. In this case $T_1(\alpha) = \frac{\alpha-1}{2}C_1^{-1} - D_1$ and $\frac{\alpha+1}{2}C_1^{-1}$ will be almost optimal. Here $\bar{D}_1 = L_1^T S_J^{-1} L_1, D_1 = M_1^T S_J^{-1} M_1$ and $C_1 = L_1^T A^{-1} L_1 = \bar{D}_1^{-1} - D_1$ denote the reduced problems. After X_1 is chosen as approximation to $T_1(\alpha)$ the modified S is defined in (6.8) by $S = S_J + L_1X_1L_1^* + M_1X_1^{-1}M_1^*$. Typically this matrix is no longer block diagonal even if X_1 is block diagonal. To get a block diagonal matrix S we have to use a block diagonal matrix $X = NN^*$ and $S = S_J + LXL^* + MX^{-1}M^*$ for the original unreduced problem. But these matrices have larger size. A natural choice will be to approximate $U_1T_1(\alpha)U_1^T + U_2U_2^T$ or $\frac{\alpha+1}{2}U_1C_1^{-1}U_1^T + U_2U_2^T$ by a block diagonal matrix X.
In order to reflect the rank deficiency of L by this choice we can utilize the nested divide & conquer method from Chapter 3. By the nested divide & conquer method from Chapter 3 we replace the initial given splitting $A = S - FF^T$ by a new splitting $A = S_1 - F_1F_1^T$ where

S_1, F_1 are obtained from S, F in the following way. If $V = \left[\tilde{V}, \hat{V}\right]$ is orthogonal, then we have

$$A = S - FF^T = \underbrace{(S - F\tilde{V}\tilde{V}^T F^T)}_{S_1} - \underbrace{F\hat{V}}_{F_1}\underbrace{\hat{V}^T F^T}_{F_1^T}.$$

By Lemma 3.18 the corresponding new coupling system $S_{c,1}$ can be obtained from S_c by taking the Schur–complement of $V^T S_c V$. In terms of the inverses we have that $S_{c,1}^{-1} = \hat{V}^T S_c^{-1}\hat{V}$. On the other hand, by (6.21) we have for the initial problem $NS_c^{-1}N^* = (X + D)C(X+D)+(X+D)$. The reduced problem will be $(X_1+D_1)C_1(X_1+D_1)+(X_1+D_1)$. But this matrix coincides with $U_1^* N S_c^{-1} N^* U_1$. From this it follows that V has to be chosen such that the columns of $\hat{V}$ span the same space as the columns of N^*U_1, i.e., the columns of $\tilde{V}$ have to span the same space as the columns of $N^{-1}U_2$. $\tilde{V}$ can be computed by performing a QR decomposition,[41],pp.211ff, of $N^{-1}\tilde{U}$.

In our example the additional amount consists of computing an initial rank 2 update after X has been chosen. In other words, in the singular case we can proceed analogously to the nonsingular case if this modification is followed by a well–chosen initial low rank update.

The arguments used here can be used as well when dividing the graph into more than four parts. To illustrate the difference between the initial coupling system S_c and the coupling system obtained by the additional low rank update we will compare their condition numbers for $p = 4, 16, 64$ and $N = 32, 64, 128$ for X obtained by Algorithm 6.89, 6.90.

Condition Number $\kappa_2(S_{c,opt})$

without low rank update

$p \backslash N$	*32*	*64*	*128*
4	$5.3{\cdot}10^1$	$8.7{\cdot}10^1$	$1.5{\cdot}10^2$
16	—	$1.7{\cdot}10^2$	$2.7{\cdot}10^2$
64	—	—	$6.5{\cdot}10^2$

after low rank update

$p \backslash N$	*32*	*64*	*128*
4	$1.2{\cdot}10^1$	$2.0{\cdot}10^1$	$3.5{\cdot}10^1$
16	—	$3.9{\cdot}10^1$	$6.3{\cdot}10^1$
64	—	—	$1.5{\cdot}10^2$

For $S_{c,fo}$ we obtain condition numbers which are relatively close to those of $S_{c,opt}$

Condition Number $\kappa_2(S_{c,fo})$

without low rank update

$p \backslash N$	*32*	*64*	*128*
4	$4.2{\cdot}10^1$	$9.5{\cdot}10^1$	$2.2{\cdot}10^2$
16	—	$1.3{\cdot}10^2$	$2.9{\cdot}10^2$
64	—	—	$5.0{\cdot}10^2$

after low rank update

$p \backslash N$	*32*	*64*	*128*
4	$1.1{\cdot}10^1$	$1.4{\cdot}10^1$	$1.8{\cdot}10^1$
16	—	$3.4{\cdot}10^1$	$4.2{\cdot}10^1$
64	—	—	$1.3{\cdot}10^2$

The reduction in the condition number results in a small number of iterations for the cg–method for Algorithm 6.89,6.90. We compare the number of iterations with that of the corresponding block Jacobi method. for $p = 4, N = 32$, $p = 16, N = 64$ and $p = 64, N = 128$.

Number of Iteration Steps

$S_J^{-1}A$

$p\backslash N$	*32*	*64*	*128*
4	20	25	30
16	—	35	42
64	—	—	58

$S_{c,opt}$

$p\backslash N$	*32*	*64*	*128*
4	10	12	14
16	—	19	21
64	—	—	32

$S_{c,fo}$

$p\backslash N$	*32*	*64*	*128*
4	11	12	12
16	—	20	21
64	—	—	33

Although an improvement in the number of iterations for the cg–method has been made in this example, the additional overhead in computing $D, \bar{D}, \bar{D}^{-1}$ cannot be equalized by the smaller number of iterations. E.g. for $p = 64, N = 128$ up to 64 right hand sides have to be computed.

Summary

The use of modified block Jacobi splittings has turned out to effectively improve the condition number of the underlying coupling system. In some examples the condition number was extremely improved. The disadvantage so far is that the generation of X requires the solution of several right hand sides with the block diagonal part of A. If e.g. the initial matrix A is block tridiagonal with all blocks of size m, then $2m$ system with the block diagonal matrix S_J have to be solved. If the cg–method with block Jacobi preconditioning does not exceed $2m$ steps, the generation of X will be more expensive in any case. However for ill–conditioned systems or larger number p of blocks this will be typically the case. In this case the generation of X will not consume most of the computing time. As additional advantage of the computation of X we will have an explicit representation of the coupling system which reduces the computation time for a matrix vector multiplication with the coupling system as well the computation time for a direct solution of the coupling system. In this case the a larger number of iterations to solve the coupling system is less critical. This is not a problem of the symmetric case and will also occur in the general case.

If the system is not well–conditioned the use of modified block Jacobi splittings can extremely improve the properties and thus the computational costs for the cg method may be reduced not only with respect to the matrix–vector multiplication but also for the number of iteration steps.

The problem which is open in general is that this theory can only be applied to block 2–cyclic matrices. But in practice we would like to reorder the given matrix by a preprocessing step. This reordering should reduce the size of the coupling system and in general the reordered matrix will not be block 2–cyclic. This reduces the application of modified block Jacobi splittings and the theory has to be adapted for the non block 2–cyclic case. The case when the matrix C from Lemma 6.19 in Chapter 6 is singular has not been discussed in detail in Chapter 6 but by Example 9.7 we have illustrated how the theory can be generalized to this case by allowing slightly more than a block diagonal splitting.

An important observation that has been made among the numerical examples is that Algorithm 6.89 and Algorithm 6.90 behave quite similar for both, the resulting condition

number and the number of iteration steps. From this point of view the main bottle neck of Algorithm 6.89, namely the computation of the parameter α in Algorithm 6.90 can be replaced by the heuristic approach from Algorithm 6.90.
So far we have only compared the sequential flop count. For a parallel implementation one has to take care of the additional communication costs. The generation of $\bar{D}, D$ has the advantage, that the their computation can be done without communication, since several systems have to be solved with a block diagonal matrix. To perform the difference $\bar{D}^{-1} - D$ one step of local data exchange is necessary and for the scaling of X one global step of communication has to be done. In contrast to this in an iterative process local and global communication have to be done in any step. So we may have a better improvement than in the **MATLAB** computations.

9.2 Parallel Numerical Results for the Positive Definite Case

In this section we compare for several examples the cg–method, preconditioned by the block Jacobi method with the nested Divide & Conquer method. For all these examples we used the partitioning obtained by METIS [52]. For the updates we use the Lanczos Process [67],[41], pp.475ff for $S_{c,k}$ with selective reorthogonalization [51],[69],[41],pp.489–489. We can easily derive the cg–method from the Lanczos process (see e.g. [41], pp.494–497,523–524) and find during the Lanczos process an approximative $\tilde{x}_k$ to the problem $S_{c,k}x_k = b_k$. In practice we used the Lanczos process for up to 50 steps. An update is made when the criterion for the orthogonality between the Lanczos vectors signals a loss of orthogonality. From the eigenvalues that have been delivered by the Lanczos process we used all extremal eigenvalues until the smallest remaining eigenvalue was greater than 10^{-2} times than the largest remaining eigenvalue. If there are no eigenvalues satisfying this criterion, then the Lanczos process is continued.

A practical problem which arises when applying the nested divide & conquer process is that for a larger number of iterations it might be useful to generate the coupling system S_c. This occurs for example for larger numbers p of processors. The coupling system is distributed over the processors and any processor will have to compute $2n_c/p$ right hand sides in the average, where n_c is the size of the coupling system. So a natural strategy to compute S_c which is also a compromise between a direct and an iterative solution will be to compute the parts of the coupling system if the number of iterations will be more than $2n_c/p$. To have a simple heuristical criterion for this we will check after a while, namely after the first update or after n_c/p iteration steps in any further step the residual of the computed solution $\tilde{x}_k$. A first criterion obtained by the Lanczos process can be obtained from the condition number. The well–known bound [41],p.525 on the norm of the residual is

$$\|r_l\|_A \leqslant 2\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^l \|r_0\|_A.$$

Here the norm is the energy norm $\|x\|_A = \sqrt{x^* A x}$ and κ is the condition number of A. In practice the number l of steps which is needed to to have the residual less than a given tolerance times the initial residual is overestimated by this inequality. We use an additional heuristic. If l steps of the iterative method have been performed we will check how long the iterative process will take, provided that the decreasing of the residual will not be worse in the next l steps than in the first l steps. This will give us an estimate for the number of iterations for the iterative process. The disadvantage will be of course if the iterative process will converge in step $l+1$ then the computation of the parts of S_c was in vain. But for larger numbers of p (e.g.16,32,64) this additional overhead will be moderate. As a consequence we have by (7.13) and Corollary 7.16 a representation of S_c in the form

$$S_c = I - \sum_{q=1}^{p} K_q \hat{M}_q K_q^T,$$

where any $\hat{M}_q$ is precisely the part of S_c which is stored on processor q. A simple preconditioner which we can immediately get from this representation is

$$\sum_{q=1}^{p} K_q (K_q^T S_c K_q)^{-1} K_q^T,$$

By the definition of K_q in Corollary 7.16 this will be a simple overlapping block diagonal preconditioner. When having a preconditioner we can construct our low rank updates with respect to the preconditioned system instead of the original system, i.e., we use the Lanczos process for the preconditioned coupling system instead of the original coupling system.

By using the nested application of the Sherman–Morrison–Woodbury formula we obtain in principle the solution of $Ax = b$, when the small coupling system $S_{c,k}$ has been solved. In practice the solution obtained by the divide & conquer approach can be perturbed solving the small coupling system. For this reason the method is embedded into an outer cg–iteration. I.e., by k steps of the nested divide & conquer method we obtain in theory

$$A^{-1} = S_k^{-1} + S_k^{-1} F_k S_{c,k}^{-1} G_k S_k^{-1}.$$

We use the numerically computed $\tilde{A}^{-1}$ as preconditioner for A in the outer cg–iteration.

As stopping criterion for the combined update and iteration we used $\|r_k\|_2 \leqslant \sqrt{\text{eps}}\,\|r_0\|_2$, where r_k denotes the residual.

The parallel computations were performed on Parsytec GCPP parallel computer. This parallel computer is MIMD computer with distributed memory. Communication is done by message passing using the communication library of [44]. For the diagonal blocks a sparse Cholesky decomposition from SPARSPAK [40] was taken. For the off–diagonal blocks a sparse LU decomposition from MA28 [27] was used. The Cholesky decomposition of the small coupling systems was done using LAPACK[1]. For elementary vector operations the BLAS library[23] was used.

Example 9.8
Consider the matrix **LANPRO/NOS2** *from Example 9.2. Since this matrix is quite small ($n = 957$) for parallel computations, we extend this matrix. The matrix has the form*

$$\begin{pmatrix} C & B & & O \\ B^T & C & B & \\ & \ddots & \ddots & \ddots \\ O & & B^T & C \end{pmatrix}.$$

We extend this matrix by taking the corresponding block tridiagonal matrix with size $8 \cdot n, 16 \cdot n$ and $32 \cdot n$, i.e., essentially the matrix is taken up to 32 times. We first examine the block Jacobi method for this matrix.

Time [sec] needed for Cholesky decomposition of S_J

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	1.08	0.27	0.09	0.03		
$16 \cdot n$	4.45	0.50	0.28	0.09	0.04	
$32 \cdot n$	17.42	4.45	1.13	0.28	0.09	0.06

When increasing the number of processors the time required by the Cholesky decomposition decreases approximately quadratically. More important is the time and the number of iterations for the iterative solution process. The results are very disappointing. The number of iterations drastically increases with the number of processors which makes the iterative method no longer applicable to this problem.

Number of iterations needed for the cg–method

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	13	30	133	397		
$16 \cdot n$	25	57	207	614	2423	
$32 \cdot n$	19	44	179	652	3369	13973

Obviously the time for the iterative process will become huge as well when increasing the number of processors.

Time [sec] needed for the iterative solution process

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	0.39	0.51	1.65	4.61		
$16 \cdot n$	1.51	1.86	4.48	9.43	38.75	
$32 \cdot n$	2.17	2.71	6.78	15.62	63.80	385.41

As a consequence of the huge number of iterations the total amount will be essentially dominated by the iterative part for larger p.

Total amount [sec]

size(A) \ p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	1.47	0.78	1.74	4.64		
$16 \cdot n$	5.96	2.36	4.76	9.52	38.79	
$32 \cdot n$	19.59	7.16	7.91	15.90	63.89	385.47

The iterative part consumes for $p > 4$ most time for the solution process. For $p > 4$ the extremal number of solution steps makes an iterative solution impossible.

Next we study the nested divide & conquer method applied to this problem. Like for the block Jacobi method we start with the time needed for the Cholesky decomposition. We expect that there is no essential difference which is confirmed by the numerical results.

Time [sec] for the Cholesky decomposition

size(A) \ p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	1.11	0.28	0.10	0.05		
$16 \cdot n$	4.34	0.52	0.29	0.12	0.07	
$32 \cdot n$	17.92	4.34	1.11	0.32	0.13	0.16

Next we examine the number of iterations needed by the nested divide & conquer method applied to this problem. In this example the coupling system was generated after 3 steps except for $p = 2$. The reason is that the coupling system is quite small and on any processor not more than a 6×6 part of S_c is stored.

Number of steps for combined update and iteration

size(A) \ p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	7	15	34	60		
$16 \cdot n$	7	15	35	60	106	
$32 \cdot n$	9	19	45	66	137	227

Compared with the standard block Jacobi method the number of steps has been extremely reduced. During the iteration the size of S_c has been successively reduced. We give the size of the initial S_c and the final S_c in the next table.

Size of the coupling system, initially $\mapsto$ finally

size(A) \ p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	*2*$\mapsto$ *1*	*5*$\mapsto$ *2*	*12*$\mapsto$ *9*	*25*$\mapsto$ *23*		
$16 \cdot n$	*2*$\mapsto$ *1*	*5*$\mapsto$ *3*	*12*$\mapsto$ *7*	*25*$\mapsto$ *21*	*52*$\mapsto$ *47*	
$32 \cdot n$	*2*$\mapsto$ *1*	*5*$\mapsto$ *3*	*12*$\mapsto$ *7*	*25*$\mapsto$ *17*	*52*$\mapsto$ *44*	*106*$\mapsto$ *95*

Beside the lower number of iteration steps and the reduction of S_c we are interested in the computational costs.

Time [sec] needed for the combined update and iteration

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	0.23	0.24	0.44	0.84		
$16 \cdot n$	0.43	0.36	0.54	1.09	2.21	
$32 \cdot n$	1.13	0.78	0.90	1.40	3.25	7.47

Note that this time table gives the time, which includes arithmetic and communication costs. For larger p the computational costs are still dominated by the communication part, although the updates have been collected to keep the data traffic small. This can be seen, when we examine the arithmetic part of the computation.

Maximal arithmetic time [sec] for the combined update and iteration

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	0.21	0.14	0.11	0.12		
$16 \cdot n$	0.41	0.27	0.17	0.15	0.27	
$32 \cdot n$	1.08	0.66	0.46	0.28	0.36	0.86

This is not only a problem for the nested divide & conquer process. The same problem also occurs for the block Jacobi method.

Altogether we have the following total amount for the nested divide & conquer process.

Total amount [sec]

p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	1.34	0.52	0.54	0.89		
$16 \cdot n$	4.77	0.88	0.83	1.21	2.28	
$32 \cdot n$	19.05	5.12	2.01	1.72	3.38	7.63

What we can see in this example which is really ill–conditioned is that not only the number of iterations for the nested divide & conquer is much less than for the standard block Jacobi but that in addition several updates were necessary to adapt the process to the problem. The improvement made by the nested divide & conquer method is not only due to the additional preconditioner for S_c which one can see when only taking the preconditioned coupling system with no later update. Here the number of iterations also becomes huge for larger p and even worse, in several cases the iterative process did not converge!

Number of steps for iterative process

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	7	19	31	4017		
$16 \cdot n$	7	19	36	∞	∞	
$32 \cdot n$	9	24	132	∞	∞	∞

In those cases where the iterative process did not converge, the residual was decreasing for a while and then increasing again and finally the process diverged.

Time for the iterative process

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	0.23	0.16	0.26	27.64		
$16 \cdot n$	0.43	0.25	0.29	∞	∞	
$32 \cdot n$	1.13	0.55	0.89	∞	∞	∞

Beside the smaller number of iteration steps the nested divide & conquer process has another advantage. The number of iterations and thus the computing time for solving a further right hand side will be much less than for the first system. We will show this in the following tables.

Number of steps for a further right hand side

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	6	7	25	37		
$16 \cdot n$	6	9	18	38	55	
$32 \cdot n$	8	12	23	34	52	91

Time [sec] needed for a second right hand side

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	0.22	0.16	0.38	0.65		
$16 \cdot n$	0.42	0.27	0.33	0.82	1.42	
$32 \cdot n$	1.12	0.63	0.60	0.80	1.49	3.41

The problem that the communication part overlays the arithmetic part will occur for a further right hand side again.

In this example the nested divide & conquer has turned out to be a quite useful method to overcome problems when the system is really ill–conditioned and an iterative process like the block Jacobi will not converge. Of course for extremal cases like this we cannot always expect that by increasing the number of processors we will need less time since the number of iterations may rapidly increase. More important is the observation that in this example the iterative process has been stabilized using the nested divide & conquer method leading to a moderate number of iterations as well as to a reduction of the system. The initial block diagonal matrix will be successively adapted to the problem using low rank modifications. The fact that the computational costs increase for larger p may be related to the fact that the coupling system is extremely small and that the communication is quite expensive. On any processor not more then a 6×6 *block of the coupling system is stored. Thus the numerical costs will be very small compared with communication costs. Nevertheless up to* $p = 16$ *we could reduce the total amount and in contrast to the block Jacobi method the computational costs for* $p = 32, 64$ *is still moderate. Another point which is not satisfactory so far is the* *local data exchange. Currently a general purpose routine based on the graph of the initial system is used. What is missing so far is a communication routine which adapts the local data exchange required by the block graph of the initial system to the physical communication network.*

Example 9.9
The example that we will discuss is closely connected to the matrix **LANPRO/NOS1** *in Example (9.1). Since the size of this example ($n = 237$) is much too small to use it in parallel computations we will use the matrix $A = I \otimes T + T \otimes I$, where T denotes the matrix from (9.1). The resulting matrix has size $n = 56169$.*

We will examine the parallel block Jacobi method and the nested divide & conquer method for $p = 2, 4, \ldots, 64$.

We start with the block Jacobi method.

Time [sec] for Cholesky decomposition of S_J

p	*2*	*4*	*8*	*16*	*32*	*64*
	52.57	16.43	6.53	2.57	0.89	0.35

The time needed for the Cholesky decomposition decreases by a factor between 2 and 3 when increasing the number of processors. For larger p the time and the number of iterations needed for the iterative process comes more and more into account and will dominate the total amount. Unfortunately the number of iteration steps is huge.

Number of iterations needed by the cg–method

p	*2*	*4*	*8*	*16*	*32*	*64*
	43	842	2041	2577	3000	3436

Consequently the time will be high as well.

Time [sec] needed for the iterative solution process for $S_J^{-1}A$

p	*2*	*4*	*8*	*16*	*32*	*64*
	38.60	331.57	414.38	252.03	158.86	134.18

Already for $p > 2$ the iterative solution process dominates the total costs.

Total time [sec]

p	*2*	*4*	*8*	*16*	*32*	*64*
	91.17	348.00	420.91	254.60	159.75	134.53

Here for $p = 2$ the least time is needed. One important reason for this effect is that the coupling system S_c will rapidly increase in its size as the number of processors increases. In other words for the block Jacobi method the rank of $S_J - A$ is rapidly increasing. The rank will be typically 2 times more then the size of the coupling system for the nested divide & conquer process. The size of S_c for the minimal rank approach will be $84, 448, 1099, 1892, 3216, 5058$ for $p = 2, 4, 8, \ldots$ and thus the rank of $S_J - A$ will be approximately twice as much. This is a specific property of this matrix and clearly this will also affect the nested divide & conquer method.
It is clear that the time required for the Cholesky decomposition will be almost the same for the nested divide & conquer method as for the block Jacobi method.

Time [sec] for the Cholesky decomposition

p	*2*	*4*	*8*	*16*	*32*	*64*
	51.28	19.26	7.41	2.67	1.09	0.49

More important is the number of iterations and the time needed for the iterative process. Here up to the case $p = 2$ the coupling system was generated during the iteration (for $p = 4$ after the second update in step 100 and for $p > 4$ after the first update in step 50). The updates were then performed with respect to the preconditioned system.

Number of iterations needed for the combined update and iteration

p	*2*	*4*	*8*	*16*	*32*	*64*
	46	121	124	206	397	529

The number of iterations for the nested divide & conquer is already much less than for the block Jacobi method. During the nested divide & conquer process the coupling system has been reduced in its size.

Size of the coupling system, initially $\mapsto$ finally

p	*2*	*4*	*8*	*16*	*32*	*64*
	84$\mapsto$*83*	*448*$\mapsto$*443*	*1099*$\mapsto$*1093*	*1892*$\mapsto$*1884*	*3216*$\mapsto$*3198*	*5058*$\mapsto$*5025*

The fewer number of iterations will also reduce the time for the solution process.

Time [sec] needed for the combined update and iteration

p	*2*	*4*	*8*	*16*	*32*	*64*
	51.96	156.25	76.93	50.62	46.77	39.45

Like in Example 9.8 the arithmetic part will be much less for larger p. Here the situation is not so drastical as in Example 9.8.

Maximal arithmetic time [sec] for the combined update and iteration

p	*2*	*4*	*8*	*16*	*32*	*64*
	51.88	154.60	75.30	46.55	36.10	18.46

Finally we have the following total amount.

Total amount [sec]

p	*2*	*4*	*8*	*16*	*32*	*64*
	103.24	175.51	84.34	53.29	47.86	39.94

We see that the nested divide & conquer will need much less iterations and much less time than the block Jacobi method. Again the fewer number of iterations is not only a consequence of the additional use of a preconditioner for the coupling system, which can be seen when we omit later updates.

Number of iterations without later updates

p	*2*	*4*	*8*	*16*	*32*	*64*
	46	121	129	238	458	859

Time [sec] needed for the iteration without later updates

p	*2*	*4*	*8*	*16*	*32*	*64*
	51.94	157.01	76.74	53.31	49.04	53.37

Again the solution of a further right hand side is quite cheap when using the nested divide & conquer method.

Number of steps for a further right hand side

p	*2*	*4*	*8*	*16*	*32*	*64*
	44	21	54	89	114	130

Time [sec] needed for a second right hand side

p	*2*	*4*	*8*	*16*	*32*	*64*
	50.90	4.35	5.99	10.79	11.81	10.84

One problem which aggravates the parallel solution of this system is the fact that with increasing number of processors the coupling rapidly grows. This is a problem for any method working with low rank splittings. This is the reason why still for $p = 8$ *the method is only slightly faster than for* $p = 2$. *In addition the size of the distributed parts is strongly varying especially for* $p = 4$. *In this case two processors had a part of the coupling system* *which has a size larger than* 320 *while the other ones had parts of* S_c *in the order of* 100. *This effect also occurs for* $p = 8, 16$ *but there the different sizes of the parts of* S_c *are not so strongly varying as for* $p = 4$. *For* $p = 64$ *the coupling system already has size larger than* 5000 *while the whole system has a size of* $n = 56169$. *In addition it is ill–conditioned which is stated by the enormous number of iterations needed by the block Jacobi method. Consequently the improvement in the computational time will not be so strong when increasing the number of processors. With respect to this specific problems the improvement which is made using the nested divide & conquer approach is still quite acceptable.*

Example 9.10
We consider the matrix **BCSSTRUC1/BCSSTK09** *from Example 9.4. Analogously to Example 9.8 we extend the matrix by essentially taking the matrix* 8, 16 *and* 32 *times.*

We start this example by examining the block Jacobi method for this problem.

Time [sec] for Cholesky decomposition

size(A) \ p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	6.46	2.86	0.90	0.29		
$16 \cdot n$	10.62	5.91	2.69	1.00	0.29	
$32 \cdot n$	32.21	14.41	6.35	3.23	1.09	0.33

For larger p *the number of iterations for the cg–method will consume most of the computational costs since the number of iterations will increase.*

Number of iterations needed for the cg–method

size(A) \ p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	62	133	224	448		
$16 \cdot n$	58	136	246	322	514	
$32 \cdot n$	52	128	209	228	256	478

Since the number of iterations will increase the time will at least increase in those cases where the number of iterations drastically increases from p *to* $2p$.

Time [sec] needed for the iterative solution process

size(A) \ p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	10.29	10.66	8.43	9.83		
$16 \cdot n$	17.74	21.86	20.37	13.34	13.62	
$32 \cdot n$	38.68	45.10	35.69	20.44	12.22	16.56

Total amount [sec]

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	16.75	13.52	9.33	10.12		
$16 \cdot n$	28.36	27.77	23.06	14.34	13.91	
$32 \cdot n$	70.89	59.51	42.04	23.67	13.31	16.89

For the total amount increasing the number of processors will make the block Jacobi faster up to the last but one processor that has been taken here. But finally for $8 \cdot n, p = 16$ *and for* $32 \cdot n, p = 64$ *the number of iterations will increase too much.*

Next we study the nested divide & conquer method applied to this problem. Like for the block Jacobi method we start with the time needed for the Cholesky decomposition which will be almost the same as for the block Jacobi method

Time [sec] for the Cholesky decomposition

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	5.19	2.58	1.06	0.37		
$16 \cdot n$	11.43	5.66	2.83	1.16	0.40	
$32 \cdot n$	36.03	10.23	7.02	2.89	1.13	0.46

Next we examine the number of iterations needed by the nested divide & conquer method applied to this problem. Here after the first update (for $p = 2$ *after 33-36 steps, in the other cases after 50 steps) the coupling system is generated and the update is made with respect to the preconditioned system.*

Number of steps for combined update and iteration

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	39	54	63	124		
$16 \cdot n$	36	53	58	64	206	
$32 \cdot n$	36	51	56	58	64	241

Due to the generation of the parts of S_c *the improvement in the number of iterations is partially weakened.*

Time [sec] needed for the combined update and iteration

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	13.74	10.71	5.39	4.46		
$16 \cdot n$	26.88	23.13	12.48	5.58	8.89	
$32 \cdot n$	70.45	44.64	28.01	12.48	5.75	12.37

The pure arithmetic part of the computation will be much less for larger p.

Maximal time [sec] needed for arithmetic part

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	13.66	10.35	4.92	2.61		
$16 \cdot n$	26.81	22.00	11.70	4.80	3.80	
$32 \cdot n$	70.38	43.95	26.45	11.87	4.84	5.02

Here the increasing time for larger p obviously is related to the increasing amount for the communication. As mentioned in Example 9.8, one problem is the current realization of the local data exchange, which is not yet satisfactory. Like in Example 9.8 this problem also occurs for the block Jacobi method.

During the iteration the size of S_c has been successively reduced. We give the size of the initial S_c and the final S_c in the next table.

Size of the coupling system, initially $\mapsto$ finally

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	*72* $\mapsto$ *72*	*191* $\mapsto$ *190*	*472* $\mapsto$ *469*	*987* $\mapsto$ *978*		
$16 \cdot n$	*66* $\mapsto$ *66*	*198* $\mapsto$ *196*	*482* $\mapsto$ *479*	*996* $\mapsto$ *993*	*2078* $\mapsto$ *2066*	
$32 \cdot n$	*66* $\mapsto$ *66*	*192* $\mapsto$ *192*	*461* $\mapsto$ *456*	*958* $\mapsto$ *955*	*1946* $\mapsto$ *1943*	*4224* $\mapsto$ *4209*

Up to the final number p of processors that have been used the reduction of S_c is quite small. For the final number p of processors that have been used the number of iterations is larger and consequently S_c will be more reduced in its size. To summarize the nested divide & conquer we have the following total amount.

Total amount [sec]

p	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	18.93	13.29	6.45	4.83		
$16 \cdot n$	38.31	28.79	15.30	6.74	9.29	
$32 \cdot n$	106.48	54.87	35.03	15.37	6.88	12.83

Up to the case $p = 2$ where the heuristic for the generation of S_c has slowed down the nested divide & conquer strategy the total amount for the nested divide & conquer is less than for the block Jacobi method and the reduction of S_c is moderate except for the case of the finally used p, but there the number of iterations also clearly increases.
Here only slight reductions were necessary and this is the reason why without using updates the time does not essentially differ from nested divide & conquer process. In fact for $16 \cdot n$, $p = 32$ and $32 \cdot n$, $p = 64$ it needs slightly less iterations.

Number of steps for the iteration with no further updates

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	39	54	63	146		
$16 \cdot n$	36	54	58	64	197	
$32 \cdot n$	36	51	56	58	64	233

Time [sec] needed for the iteration with no further updates

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	13.74	10.61	5.36	4.81		
$16 \cdot n$	26.88	22.62	12.47	5.58	7.70	
$32 \cdot n$	70.41	44.67	28.07	12.49	5.74	10.53

The situation drastically changes when we replace the initial matrix A by a shifted matrix $A - \mu I$. The first 10 digits of μ coincide with those of the smallest eigenvalue of A. Again we will examine both methods for the shifted matrix which has the same eigenvectors and distribution over the processors as the original system.

For the block Jacobi the number of iterations and the computational costs will increase too much for a sensible application to this problem.

Number of iterations needed for the cg–method

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	265	643	1407	1800		
$16 \cdot n$	238	696	1428	3335	4149	
$32 \cdot n$	139	446	1094	2290	4753	9112

The drastical increase in the number of iterations will be reflected in the computational time leading to a long time the more we increase the number of processors.

Time [sec] needed for the iterative solution process

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	44.27	51.68	53.26	39.60		
$16 \cdot n$	73.39	112.00	118.62	138.39	113.80	
$32 \cdot n$	103.62	157.34	187.19	205.55	222.91	316.66

Of course the nested divide & conquer will also be affected by this ill–conditioned matrix. But the impact of this ill–conditioned matrix is much weaker.

Number of steps for combined update and iteration

size(A) \ *p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	42	66	96	194		
$16 \cdot n$	40	61	90	122	365	
$32 \cdot n$	39	61	92	113	162	603

Time [sec] needed for the combined update and iteration

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	14.67	11.60	6.26	6.72		
$16 \cdot n$	28.82	23.82	13.49	7.39	16.24	
$32 \cdot n$	75.39	46.63	29.53	14.25	9.60	34.47

Here the communication overlays the arithmetic part much more than in the unshifted case, which can be seen, when considering only the maximal arithmetic costs.

Maximal arithmetic time [sec] for the combined update and iteration

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	14.56	11.10	5.32	3.41		
$16 \cdot n$	28.73	22.55	12.27	5.41	6.28	
$32 \cdot n$	75.30	45.83	27.48	12.60	5.85	12.32

In contrast to the unshifted matrix now S_c *is much more reduced in its size.*

Size of the coupling system, initially $\mapsto$ *finally*

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	*72* $\mapsto$ *71*	*191* $\mapsto$ *186*	*472* $\mapsto$ *465*	*987* $\mapsto$ *977*		
$16 \cdot n$	*66* $\mapsto$ *65*	*198* $\mapsto$ *194*	*482* $\mapsto$ *475*	*996* $\mapsto$ *990*	*2078* $\mapsto$ *2058*	
$32 \cdot n$	*66* $\mapsto$ *65*	*192* $\mapsto$ *188*	*461* $\mapsto$ *455*	*958* $\mapsto$ *950*	*1946* $\mapsto$ *1941*	*4224* $\mapsto$ *4182*

The reduction in the total amount compared with the block Jacobi is not only a consequence of the additional preconditioner, which can be seen when we do no further updates. Here for $16 \cdot n$ *and* $p = 32$ *as well as for* $32 \cdot n$ *and* $p = 64$ *the iteration did not converge. In both cases the residual was decreasing for a while and then stagnating. A similar effect occurs for* $32 \cdot n$ *and* $p = 16$*. Here the iteration was stagnating for a long time and then slowly converging. This is the reason for the enormous time and number of steps in this case.*

Number of steps for the iteration with no further updates

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	41	117	102	515		
$16 \cdot n$	36	85	111	358	∞	
$32 \cdot n$	39	87	155	14722	205	∞

Time [sec] needed for the iteration with no further updates

size(A) $\backslash p$	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	14.01	11.84	6.31	14.88		
$16 \cdot n$	28.84	23.26	13.84	13.38	∞	
$32 \cdot n$	71.84	45.48	30.59	377.29	10.82	∞

What we can see is that in this example which is really ill–conditioned not only the number of iterations for the nested divide & conquer is much less than for the standard block Jacobi but that in addition several updates were necessary to adapt the process to the problem. Here the nested divide & conquer process will also slow down for the last p that has been used but the number of iterations is still much below the size of S_c and in any case S_c has been successively reduced.
When solving systems with further right hand sides this will lead to a few number of iterations.

Number of steps for a further right hand side

*size(A)**p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	6	11	26	71		
$16 \cdot n$	6	9	20	53	117	
$32 \cdot n$	6	9	25	45	78	129

Time [sec] needed for a further right hand side

*size(A)**p*	*2*	*4*	*8*	*16*	*32*	*64*
$8 \cdot n$	1.57	0.95	0.92	2.31		
$16 \cdot n$	3.27	1.79	1.24	1.83	5.39	
$32 \cdot n$	8.40	3.21	2.49	2.07	3.26	8.47

Again the communication part will overlay the arithmetic part for larger p.

In this example the nested divide & conquer process has turned out to be competitive to the block Jacobi method for the unshifted initial matrix and it has been extremely superior for the ill–conditioned shifted matrix. While the block Jacobi will no longer be applicable to this problem the nested divide & conquer will be much less affected by the bad condition of A. But for the shifted matrix more updates are necessary and a bigger part of S_c is directly solved by the reduction. Of course the nested divide & conquer will also slow down for the final p that has been used, but in this case the system is much more reduced leading more and more to a direct solution of S_c. While for the unshifted matrix the nested divide & conquer will almost coincide with the version with no further updates, the situation drastically changes for the shifted matrix, where without using updates the convergence extremely slows down or will not converge at all.

Summary

The use of the nested divide & conquer method has turned out to be a useful method to overcome problems which occur when an iterative method will need a large number of iterations. We have illustrated for the positive definite case how this method can be efficiently implemented on a parallel computer. When the cg–method with block Jacobi preconditioning slows down and becomes useless for parallel computations the concept of making a compromise between an iterative and direct solution will be clearly superior. We have illustrated that this is not only the fact that for larger number of iterations the coupling system is build and an additional preconditioner is available but in addition the nested divide & conquer strategy will generate more and more updates the longer the iteration takes generating adaptively a preconditioner for the initial system and reducing the coupling system at the same time. And while without using the updates in theses cases the iterative process slows down or does not converge any more we have at least a reduction in the size of the coupling system and the numerical experiments have shown that in this case the nested divide & conquer will need much less iterations. An additional effect when using the nested divide & conquer is that solving further right hand sides will be much cheaper than for the first right hand side, since due to the updates the number of iterations will become smaller and smaller.

What we not necessarily get by this approach is a scale–up when using this method as black box solver. The problem is that the nested divide & conquer still needs the eigenvector information from the (preconditioned) coupling system. In theory it is possible to use larger number of updates in any step than in our numerical experiments but this would typically slow down the iterative process, since the updates do not contain more essential information for the iterative process but require more computational costs, more storage and more communication.

9.3 Some Unsymmetric Examples

We will now examine some unsymmetric examples. In contrast to the symmetric case here we can only work with numerical observations, since the approximation properties of the modified block Jacobi splitting from Chapter 6 have not been discussed. In addition we have to restrict ourselves to examples with nonsingular block diagonal part, but this reduces the freedom of examples that can be examined.

We will compare Algorithm 6.89,6.90 with the simple modified block Jacobi splitting where only $X = I$ is used. In addition we will compare these methods with the unmodified block Jacobi splitting. Like in the symmetric case the relation between the number of blocks p and the size n of the system together with the size of the blocks will restrict the freedom in choosing p. In order to investigate whether we can make an improvement when using modified block Jacobi splittings we will examine the following topics.

1. the condition number κ_2 of $S_{c,J}$, $S_{c,mod}$, $S_{c,fm}$, $S_{c,I}$. Here $S_{c,J}$ is the coupling system when using the Sherman–Morrison–Woodbury formula for the block Jacobi splitting. By Corollary 2.6 we have $AS_J^{-1}F = FS_{c,J}$ where F denotes the matrix which has unit vectors as columns with respect to the nonzero rows of $W = S - A$, i.e., we have $W = F(F^TW)$. The reason for using $S_{c,J}$ instead of AS_J^{-1} is that its size is many times smaller than that of AS_J^{-1} and this simplifies estimating the condition number of $S_{c,J}$. $S_{c,mod}$ denotes the coupling system of Algorithm 6.89, $S_{c,fm}$ is the coupling system obtained by Algorithm 6.90. Finally $S_{c,I}$ denotes the coupling systems of the modified block Jacobi-splitting for $X = I$. For estimating the condition numbers we used **MATLAB** 's 'condest' function.

2. the eigenvalue distribution of these matrices.
3. the number of sequential floating point operations(flops) for the LU–decomposition versus the flops required by the generation of X in Algorithm 6.90.
4. the number of flops for the iterative solution process for the block Jacobi method versus the number of flops required by Algorithm 6.90 and for $S_{c,I}$ when using the GMRES [75] method.

To examine the condition number of the coupling system will be important when applying a direct solution method to S_c as well as for the iterative solution of S_c. For the iterative solution of S_c using GMRES a moderate condition number allows the use of the modified Gram–Schmidt process [41], pp.218 for the reorthogonalization during the GMRES iteration [71], while otherwise reorthogonalization using Householder transformation [41],pp.211ff is preferred.

Of course a small condition number and an improved eigenvalue distribution may have nothing to do with a small number of iterations in general [42]. We will apply the full GMRES method using Householder reorthogonalization to the block Jacobi method as well to $S_{c,I}, S_{c,mod}, S_{c,fm}$.

As stopping criterion for the solution process we will use $\|r_k\|_2 \leqslant \sqrt{\text{eps}}\,\|r_0\|_2$, where r_k is the residual in step k. Here eps $\approx 2.2204 \cdot 10^{-16}$. The iterative solution is performed ten times for random right hand sides and finally the average is taken. As initial guess we will choose $x_0 = 0$.
Again the computations were carried out using **MATLAB** [60]. For the computation of the parameter α in Algorithm 6.89 we used **MATLAB** 's 'eig' function.
Note that by computing $\bar{D}, D$ from Lemma 6.19 and X we get an explicit representation of S_c in (6.9). This will reduce the number of flops when applying the GMRES–method to S_c.

In practice the GMRES method is only used for a limited number, say k, of steps. Then the iteration is restarted with the computed solution. The reason for using restarts usually is memory requirement as well computational costs. For detail we refer to [74]. For our problem here the memory requirement is not so critical, since the iteration is only performed for the coupling system, which is typically much smaller than the initial matrix. In practice it may happen that when using a restarted version of the GMRES method the number of iteration steps can drastically increase. Clearly increasing the number of GMRES steps may reduce this danger. But in practice we do not know if increasing k will effectively reduce the computational costs and in addition we do not know how large k has to be chosen. This is the point where we can introduce the nested divide & conquer process.

The nested divide & conquer method has been constructed to make a compromise between a direct solution of S_c and an iterative solution for S_c. Unfortunately the problem is that in general it is not known what orthogonal transformations should be used for the update procedure. In the symmetric positive definite case we have shown in Theorem 3.31 that skillful linear combinations of eigenvectors of S_c are optimal in the sense of quadratic forms. What one can do in the general case is to use some of the approximate eigenvectors from the underlying Arnoldi process to generate updates. Strategies concerning approximate eigenvectors have been used in several papers, see e.g. [62], [53], [73]. In [53] the initial system has been preconditioned by a product of rank 1 transformations. The transformations are based on eigenvalue translations. The idea in [53] was to transform the spread eigenvalues into a vicinity of 1. Spread eigenvalues are eigenvalues which are far away from the remaining eigenvalues.

In [62] eigenvectors corresponding with small modulus have been used to fill up the Krylov space over which the residual is minimized. In [62] it has been shown that approximate eigenvectors of eigenvalues with small modulus will have an essential influence on the convergence even if the these approximate eigenvectors are not quite exact.

Here we will give a simple heuristic for the choice of updates. Note that it is not our aim to present an optimal strategy but to illustrate that updates can be used to make a compromise between a direct and iterative solution of S_c. There may exist several strategies which are better. In contrast to [62], [53], [73] we will use this heuristic to perform an update and to explicitly reduce the size of the coupling system and successively modify the initial splitting. Even if the update strategy will not improve the iterative process we will finally end in a direct method.
The heuristic to create the update is described as follows. Using the GMRES method for a matrix A we obtain the equation

$$AQ_k = Q_{k+1}\hat{H}_k,$$

where $Q_l = [q_1, \dots, q_l]$, $l = k, k+1$ consists of mutually orthogonal column vectors and a $(k+1) \times k$ upper Hessenberg matrix

$$\hat{H}_k = \begin{pmatrix} h_{11} & \cdots & \cdots & h_{1k} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{kk} \\ 0 & & & h_{k+1,k} \end{pmatrix} = \begin{pmatrix} H_k \\ 0 \cdots 0 \; h_{k+1,k} \end{pmatrix}$$

In [62] it is suggested to use the generalized eigenvalue problem

$$H^T x = \frac{1}{\lambda} H^T H x \tag{9.11}$$

for the computation of eigenvalues close to zero. The author points out that this generalized eigenvalue problem is well suited to find eigenvalues of A close to zero. Here we will use this eigenvalue problem to define a simple heuristic update strategy. First of all we will determine isolated eigenvalues. These are all eigenvalues of (9.11) whose distance to the remaining eigenvalue is larger than the average. All eigenvalues of (9.11) can be located in a rectangle. Next we will define from these isolated eigenvalues those ones as spread eigenvalues which do not lie in a rectangle of half length and half width of the original rectangle. Finally all eigenvalues which have modulus less than a given tolerance (here 10^{-2}) times the remaining ones are selected as small eigenvalues in modulus.
By this heuristic we select spread eigenvalues and small eigenvalues from the spectrum of

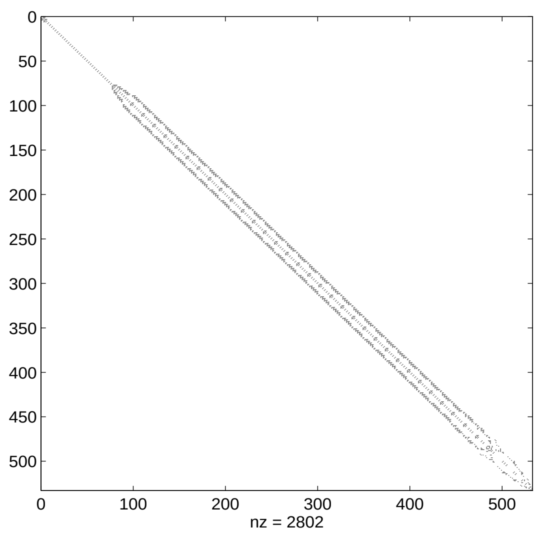
(9.11) to construct a low rank update. As update we choose the corresponding approximate invariant subspace which is spanned by the corresponding eigenvectors. I.e. If V denotes the eigenvector matrix of (9.11) corresponding to our selected eigenvalues, then a QR-decomposition of Q_kV gives us our update. Clearly this is only a heuristic. In [62] eigenvectors have been used to augment the corresponding Krylov subspace, in [53] spread eigenvalues have been translated using pole placement.

In contrast to these methods here we take out spread and small eigenvalues only from the small coupling system. By explicitly reducing the coupling system of small size the initial system will be adaptively modified by a low rank leading to an successively adapted preconditioner.

We will examine the examples for a restart $k = 30$.

Firstly we will consider some examples from the Harwell–Boeing sparse matrix collection [28]. The test matrices can be accessed via anonymous ftp from **ftp.orion.cerfacs.fr**.

Example 9.12
The matrix **PORES/PORES3** *has size is* $n = 532$ *and it is block tridiagonal with blocks of size* $\leqslant 12 \times 12$*. In this representation the matrix has been reordered using* **MATLAB** *'s 'symrcm' function which implements the reverse Cuthill–McKee ordering. Its pattern is illustrated in the picture on the right hand side. We will examine this matrix for various number of blocks* $p = 2, 4, 8, 16$.



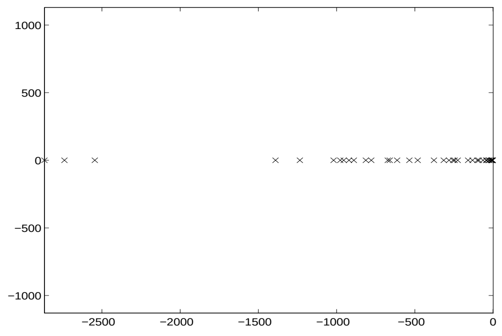
First of all we will compare the condition numbers of $S_{c,J}, S_{c,I}, S_{c,mod}$ *and* $S_{c,fm}$.

Condition Number

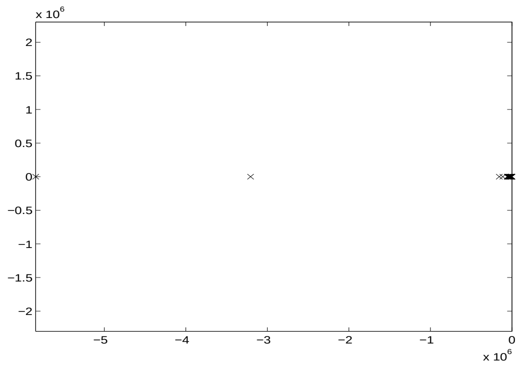
$\kappa_2 \backslash p$	*2*	*4*	*8*	*16*
$\kappa_2(S_{c,I})$	$1.3{\cdot}10^3$	$8.1{\cdot}10^3$	$1.8{\cdot}10^6$	$1.8{\cdot}10^7$
$\kappa_2(S_{c,mod})$	$1.8{\cdot}10^0$	$4.1{\cdot}10^1$	$2.6{\cdot}10^2$	$1.0{\cdot}10^4$
$\kappa_2(S_{c,fm})$	$1.0{\cdot}10^0$	$4.1{\cdot}10^1$	$2.6{\cdot}10^2$	$1.0{\cdot}10^4$
$\kappa_2(S_{c,J})$	$4.6{\cdot}10^2$	$1.4{\cdot}10^4$	$2.3{\cdot}10^4$	$4.9{\cdot}10^4$

The improvement in the condition number that has been made using $S_{c,mod}, S_{c,fm}$ *instead of* $S_{c,I}$ *is quite large. The extremely improved condition number will be useful for an iterative solution of* S_c *as well as for a direct solution of* S_c*. Clearly a small condition number will not at all be sufficient to ensure that an unsymmetric iteration like GMRES will lead to a* *small number of iterations, but an ill–conditioned system will be typically much worse. Next we will compare the eigenvalue distribution for* $p = 4, 16$*. The matrices are multiplied by a scalar such that the smallest eigenvalue in modulus has absolute value 1.*

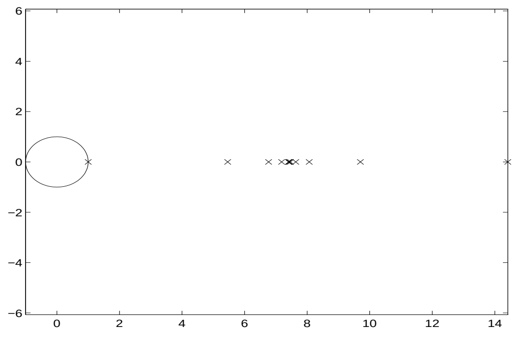
Eigenvalues of $S_{c,I} \cdot 3.7 \cdot 10^8,\ p = 4$



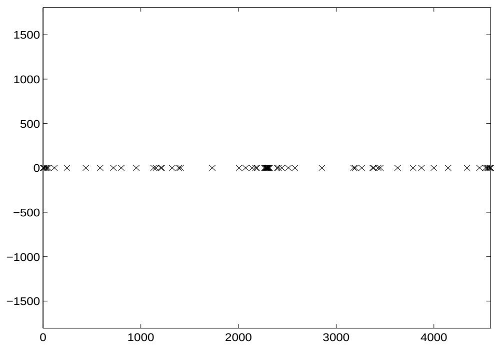
Eigenvalues of $S_{c,I} \cdot 1.3 \cdot 10^9,\ p = 16$



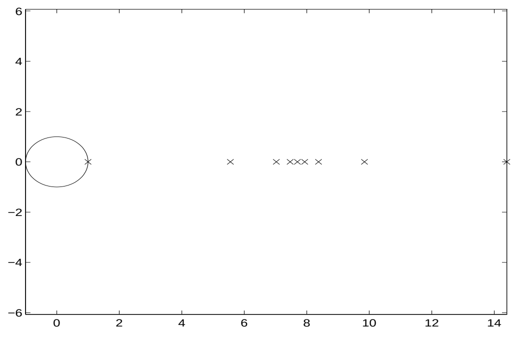
Eigenvalues of $S_{c,mod} \cdot 1.8 \cdot 10^9,\ p = 4$



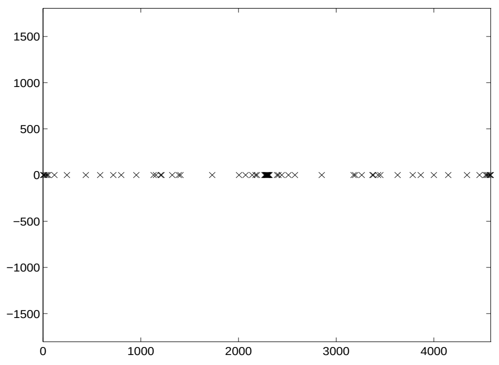
Eigenvalues of $S_{c,mod} \cdot 5.7 \cdot 10^{11},\ p = 16$



Eigenvalues of $S_{c,fm} \cdot 1.9 \cdot 10^9,\ p = 4$



Eigenvalues of $S_{c,fm} \cdot 5.7 \cdot 10^{11},\ p = 16$



The eigenvalues of $S_{c,I}$ *are widely distributed for* $p = 4$ *and even worse for* $p = 16$*. Already for* $p = 4$ *the largest eigenvalue in modulus has a modulus of more than* 2500 *while for* $S_{c,mod}$*,* $S_{c,fm}$ *the largest eigenvalue in modulus is approximately 14. For* $p = 16$ *the situation is even worse. The largest eigenvalue of* $S_{c,I}$ *in modulus has a real part of approximately* $5 \cdot 10^6$ *while for* $S_{c,mod}, S_{c,fm}$ *this is approximately* 5000*. Clearly we cannot expect that the* *improvement on the coupling system is for* $p = 16$ *as well as for* $p = 4$ *since by construction of* X *from Algorithm 6.89, 6.90* X *is only allowed to be block diagonal. The eigenvalue distribution of* $S_{c,mod}$ *and* $S_{c,fm}$ *are quite close to each other. This was also observed in the other numerical examples.*

Next we will examine the number of iterations using the full GMRES method. The iteration for $S_{c,I}$,$S_{c,mod}$ *and* $S_{c,fm}$ *will be compared with the GMRES method using* S_J *as preconditioner for* A.

Number of Iteration Steps of full GMRES

it. $\backslash p$	2	4	8	16
$S_{c,I}$	13	58	99	199
$S_{c,mod}$	6	17	31	107
$S_{c,fm}$	4	16	31	107
$S_J^{-1}A$	15	36	63	120

The number of iteration steps has been effectively reduced using the modified coupling systems $S_{c,fm}$, $S_{c,mod}$. *The reduction in the number of iterations has the disadvantage that the computation of* X *has to be taken into account. For this reason we will compare the number of* flops *(for* $S_{c,mod}$ *we will skip the number of* flops , *since the calculation of the optimal parameter* α *will be typically quite expensive).*

flops *for LU Decomposition Versus the Generation of* X

flops $\backslash p$	2	4	8	16
LU decomposition $S, X = I$	$3.8{\cdot}10^5$	$3.4{\cdot}10^5$	$3.0{\cdot}10^5$	$2.2{\cdot}10^5$
Alg. 6.90 $S_{c,fm}$	$5.7{\cdot}10^5$	$8.2{\cdot}10^5$	$1.1{\cdot}10^6$	$1.4{\cdot}10^6$
LU decomposition S_J	$3.7{\cdot}10^5$	$3.2{\cdot}10^5$	$2.7{\cdot}10^5$	$1.6{\cdot}10^5$

Of course the computation of X *will be much more expensive than a pure LU decomposition, since for the generation of* X *a block diagonal system with up to* 24 *right hand sides has to be solved.*

flops *for the Solution Process*

flops $\backslash p$	2	4	8	16
$S_{c,I}$	$3.0{\cdot}10^5$	$1.0{\cdot}10^6$	$3.1{\cdot}10^6$	$1.5{\cdot}10^7$
$S_{c,fm}$	$2.1{\cdot}10^5$	$3.0{\cdot}10^5$	$6.6{\cdot}10^5$	$5.6{\cdot}10^6$
$S_J^{-1}A$	$9.8{\cdot}10^5$	$3.8{\cdot}10^6$	$1.0{\cdot}10^7$	$3.1{\cdot}10^7$

Total amount in flops

flops $\backslash p$	2	4	8	16
$S_{c,I}$	$6.7{\cdot}10^5$	$1.4{\cdot}10^6$	$3.4{\cdot}10^6$	$1.5{\cdot}10^7$
$S_{c,fm}$	$7.8{\cdot}10^5$	$1.1{\cdot}10^6$	$1.8{\cdot}10^6$	$6.9{\cdot}10^6$
$S_J^{-1}A$	$1.3{\cdot}10^6$	$4.1{\cdot}10^6$	$1.0{\cdot}10^7$	$3.1{\cdot}10^7$

Here the modified block Jacobi splitting has turned out to be more suitable than just choosing $X = I$. When only choosing $X = I$ in this example the related coupling system becomes ill–conditioned which causes problem for the direct solution of systems with S_c as well for an iterative solution. The expensive generation of X consumes much of the improvement on S_c.

Beside the full GMRES as iterative method one typically uses a restarted GMRES as iterative process. Here we will use 30 GMRES steps before a restart is performed. Clearly this is interesting in this example for $p = 8, 16$ while for $p = 2, 4$ the number of iterations for $S_{c,fm}$ and $S_J^{-1}A$ is small. Especially the nested divide & conquer strategy can be combined with this method. For the case $X = I$, Algorithm 6.90 and the system preconditioned by the block Jacobi matrix S_J we will examine the restarted GMRES combined with low rank updates. Note that the standard block Jacobi splitting also fits into the class of low rank splittings, but the rank will not be locally minimized. The corresponding coupling system will be denoted by $S_{c,J}$.

Without our low rank update strategy we obtain the following number of iterations. For the case when the number of iterations exceeds the size of the initial system we do not continue the process anymore.

Number of Iteration Steps of GMRES(30)

it.$\backslash p$	*2*	*4*	*8*	*16*
$S_{c,I}$	*13*	*301*	$> n$	$> n$
$S_{c,fm}$	*4*	*16*	*30*	$> n$
$S_{c,J}$	*15*	*61*	$> n$	$> n$

Here none of the methods is satisfactory. Even the iteration for $S_{c,fm}$ which required the least number of iterations for the full GMRES will need more than n iterations if only 30 *steps of GMRES are used.*

The situation changes in combination with the nested divide & conquer method. Here we need the following number of steps.

Number of Iteration Steps of GMRES(30) combined with nested d& c

it.$\backslash p$	*2*	*4*	*8*	*16*
$S_{c,I}$	*13*	*87*	*241*	*391*
$S_{c,fm}$	*4*	*13*	*23*	*98*
$S_{c,J}$	*15*	*49*	*121*	*211*

Of course the number of iterations will still be larger than those of the full GMRES. But at least the number of iterations will no longer exceed the size of the system. During the nested divide & conquer process the coupling system has been successively reduced in its size. In the following table the initial and the final size of S_c are given.

Size of the coupling system, initially $\mapsto$ finally

it. $\backslash p$	*2*	*4*	*8*	*16*
$S_{c,I}$	*12* $\mapsto$ *12*	*38* $\mapsto$ *26*	*79* $\mapsto$ *49*	*161* $\mapsto$ *89*
$S_{c,fm}$	*12* $\mapsto$ *11*	*38* $\mapsto$ *37*	*79* $\mapsto$ *75*	*161* $\mapsto$ *144*
$S_{c,J}$	*24* $\mapsto$ *24*	*76* $\mapsto$ *69*	*156* $\mapsto$ *140*	*323* $\mapsto$ *288*

As expected, for $S_{c,I}$ several steps of updates are necessary but this matrix is ill–conditioned, while for the other two coupling systems the heuristic for updating S_c gives a much more moderate reduction.

The successive reduction of S_c will adaptively replace the initial block diagonal matrix by a block diagonal plus an additional low rank modification. This will have the side effect that solving system with further right hand sides will require less iterations than the first right hand side. To see this we apply all algorithms to a further right hand side.

Second right hand side of GMRES(30) combined with nested d& c

it. $\backslash p$	*2*	*4*	*8*	*16*
$S_{c,I}$	*13*	*48*	*91*	*91*
$S_{c,fm}$	*4*	*13*	*19*	*44*
$S_{c,J}$	*15*	*23*	*59*	*91*

Finally we will compare the flops *when using the nested divide & conquer for one right hand side.*

flops *for the Solution Process of GMRES(30)*

flops $\backslash p$	*2*	*4*	*8*	*16*
$S_{c,I}$	$3.0{\cdot}10^5$	$8.9{\cdot}10^6$	—	—
$S_{c,fm}$	$2.1{\cdot}10^5$	$3.0{\cdot}10^5$	$6.4{\cdot}10^5$	—
$S_{c,J}$	$3.4{\cdot}10^5$	$1.9{\cdot}10^6$	—	—

flops *for the Solution Process of GMRES(30) combined with nested d& c*

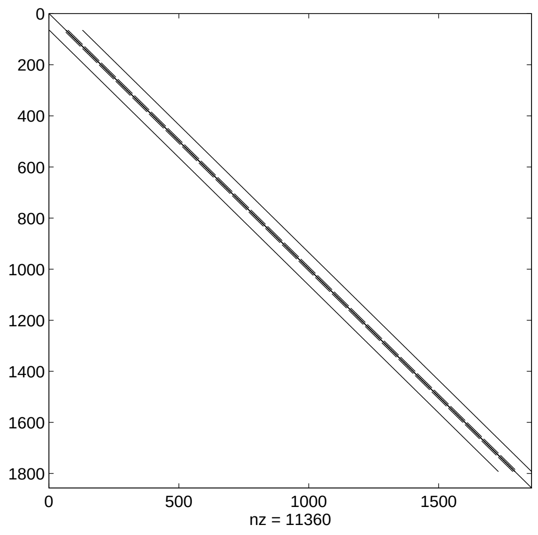
flops $\backslash p$	*2*	*4*	*8*	*16*
$S_{c,I}$	$3.0{\cdot}10^5$	$9.1{\cdot}10^6$	$3.3{\cdot}10^7$	$9.2{\cdot}10^7$
$S_{c,fm}$	$2.1{\cdot}10^5$	$3.9{\cdot}10^5$	$1.5{\cdot}10^6$	$1.4{\cdot}10^7$
$S_{c,J}$	$3.4{\cdot}10^5$	$4.6{\cdot}10^6$	$1.7{\cdot}10^7$	$5.7{\cdot}10^7$

Since the iteration steps for the GMRES(30) exceeds several times the size of the system, we can only partially compare it with the use of the nested divide & conquer strategy.
Much more important here is the observation that using the nested divide & conquer strategy the iterative process becomes more and more a direct process the longer the number of iterations will be, since the size of the coupling system will be reduced. Compared with the

GMRES(30) which does not use updates, the combination with the nested divide & conquer ensures that the process will terminate after a moderate number of steps. Note that in this example we have $n = 532$ and without the updates the iterative process for $p = 16$ will take more than n steps for all three coupling systems!

Example 9.13

*The matrix **WATT/WATT1** has size is $n = 1856$ and it is block tridiagonal with blocks of size $\leqslant 64 \times 64$. Its pattern is illustrated in the picture on the right hand side. We will examine this matrix for various number of blocks $p = 2, 4, 8$.*



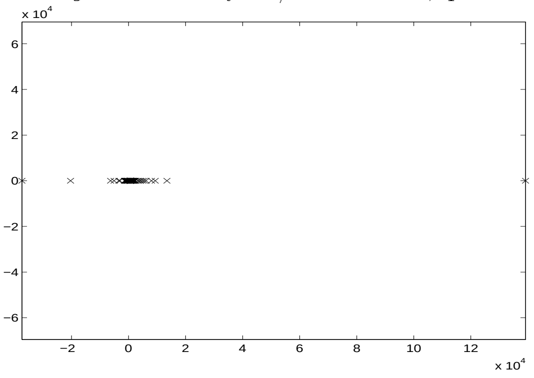
First of all we will compare the condition numbers of $S_{c,J}, S_{c,I}, S_{c,mod}$ and $S_{c,fm}$.

Condition Number

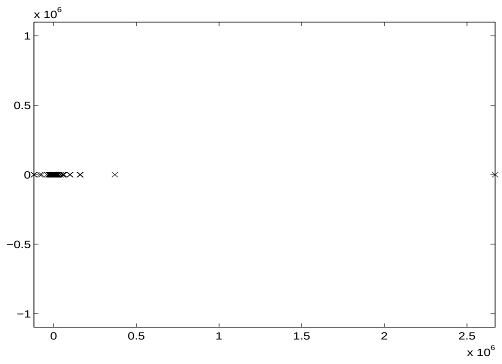
$\kappa_2 \setminus p$	*2*	*4*	*8*
$\kappa_2(S_{c,I})$	$4.1{\cdot}10^7$	$3.9{\cdot}10^8$	$2.6{\cdot}10^9$
$\kappa_2(S_{c,mod})$	$2.1{\cdot}10^0$	$4.7{\cdot}10^1$	$1.8{\cdot}10^2$
$\kappa_2(S_{c,fm})$	$1.0{\cdot}10^0$	$4.5{\cdot}10^1$	$1.8{\cdot}10^2$
$\kappa_2(S_{c,J})$	$7.0{\cdot}10^1$	$1.3{\cdot}10^2$	$2.5{\cdot}10^2$

The improvement of the condition number of $S_{c,mod}, S_{c,fm}$ compared with that of $S_{c,I}$ is remarkable. Algorithm 6.89 as well as Algorithm 6.90 extremely improve the condition number. Next we will compare the eigenvalue distribution for $p = 4, 8$. The matrices a multiplied by a scalar such that the smallest eigenvalue in modulus has absolute value 1.

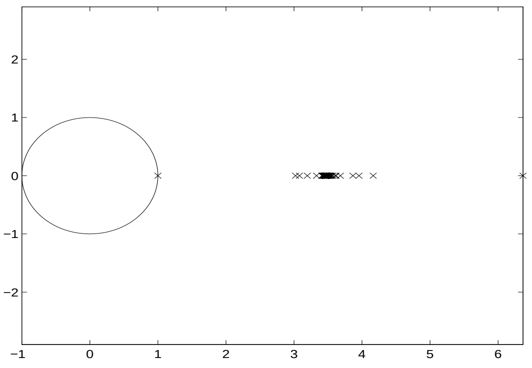
Eigenvalues of $S_{c,I} \cdot 4.3 \cdot 10^2$, $p = 4$



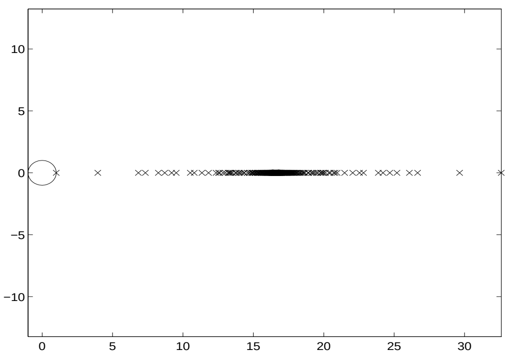
Eigenvalues of $S_{c,I} \cdot 4.3 \cdot 10^3$, $p = 8$



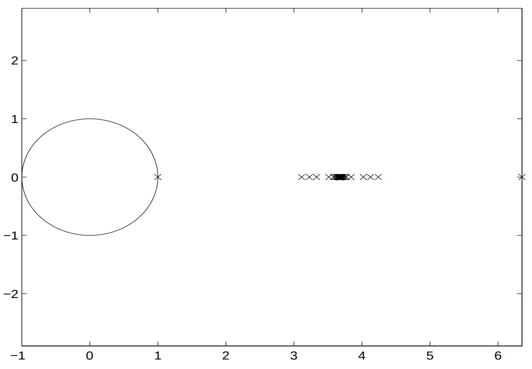
Eigenvalues of $S_{c,mod} \cdot 2.3 \cdot 10^7,\ p = 4$



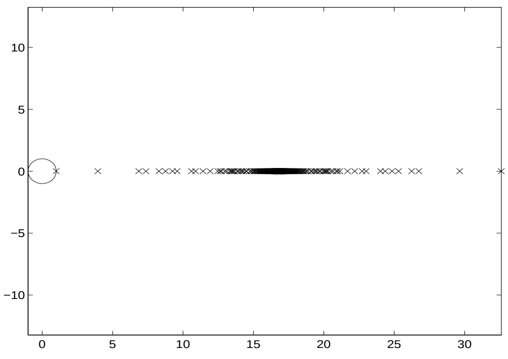
Eigenvalues of $S_{c,mod} \cdot 3.8 \cdot 10^7,\ p = 8$



Eigenvalues of $S_{c,fm} \cdot 2.1 \cdot 10^5,\ p = 4$



Eigenvalues of $S_{c,fm} \cdot 1.9 \cdot 10^5,\ p = 8$



Already for $p = 4$ *the eigenvalues of* $S_{c,I}$ *are widely distributed (watch the scaling!). They are from* $-3 \cdot 10^4$ *to* $1.3 \cdot 10^5$, *while there are still eigenvalues with modulus* 1 *which are quite small in modulus compared with the extremal eigenvalues. For* $S_{c,mod}$ *the eigenvalues relatively close to each other and no large gap is seen. The situation will be analogous for* $p = 8$, *but for both coupling systems the gaps increase. Again the eigenvalues of* $S_{c,mod}$, $S_{c,fm}$ *are quite close to each other.*

Next we will examine the number of iterations using the GMRES method. The iteration for $S_{c,I}$,$S_{c,mod}$ *and* $S_{c,fm}$ *will be compared with the GMRES method applied to the preconditioned system* $S_J^{-1}Ax = S_J^{-1}b$.

Number of Iteration Steps

it.$\backslash p$	*2*	*4*	*8*
$S_{c,I}$	*65*	*190*	*403*
$S_{c,mod}$	*8*	*15*	*30*
$S_{c,fm}$	*5*	*13*	*30*
$S_J^{-1}A$	*40*	*70*	*88*

Here the number of steps has been drastically reduced compared with the choice $X = I$. *And also the GMRES with block Jacobi preconditioning needs more steps.*
Finally we will compare the number of flops *.*

flops *for LU Decomposition Versus the Generation of X*

flops $\backslash p$		*2*	*4*	*8*
LU decomposition	$S, X = I$	$2.8{\cdot}10^7$	$2.5{\cdot}10^7$	$1.8{\cdot}10^7$
Alg. 6.90	$S_{c,fm}$	$5.2{\cdot}10^7$	$7.8{\cdot}10^7$	$1.1{\cdot}10^8$
LU decomposition	S_J	$2.8{\cdot}10^7$	$2.5{\cdot}10^7$	$1.9{\cdot}10^7$

flops *for the Solution Process*

flops $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	$2.2{\cdot}10^7$	$8.7{\cdot}10^7$	$3.4{\cdot}10^8$
$S_{c,fm}$	$1.7{\cdot}10^6$	$3.0{\cdot}10^6$	$1.0{\cdot}10^7$
$S_J^{-1}A$	$3.0{\cdot}10^7$	$6.4{\cdot}10^7$	$8.4{\cdot}10^7$

Total amount in flops

flops $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	$5.1{\cdot}10^7$	$1.1{\cdot}10^7$	$3.6{\cdot}10^8$
$S_{c,fm}$	$5.3{\cdot}10^7$	$8.1{\cdot}10^7$	$1.2{\cdot}10^8$
$S_J^{-1}A$	$5.8{\cdot}10^7$	$8.8{\cdot}10^7$	$1.0{\cdot}10^8$

For the generation of X in Algorithm 6.90 here up to 128 right hand sides have to be solved with S_J which makes the generation of X quite expensive. But without modifying the block diagonal part of S the condition number and the number of iterations grow drastically. At the end the improvement in the condition already is worth enough to use the modified block Jacobi splitting instead of just using $X = I$.

Next we will use the GMRES(30) method instead of the full GMRES method. The number of iterations will then grow drastically but again the nested divide & conquer method can be used to reduce the number of iterations as well as the size of the coupling system.

Number of Iteration Steps of GMRES(30)

it. $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	$> n$	$> n$	$> n$
$S_{c,fm}$	*5*	*14*	*30*
$S_{c,J}$	*31*	*61*	*91*

For $S_{c,fm}$ we need not apply the nested divide & conquer process since the number of iterations is still under the limit 30.

Applying the nested divide & conquer method to $S_{c,I}$ and $S_{c,J}$ will reduce the number of iterations and the coupling systems will be successively reduced.

Number of Iteration Steps of GMRES(30) combined with nested d& c

it. $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	*72*	*252*	*691*
$S_{c,J}$	*31*	*61*	*91*

Here for $S_{c,I}$ the number of iterations is still pretty large, but without using the updates we have more than $n = 1856$ iteration steps. For $S_{c,J}$ the reduction in the number of iterations is not so high, but the number of iterations is already moderate without using the nested divide & conquer process. The reduction of the size of the coupling system is given in the following table.

Size of the coupling system, initially $\mapsto$ finally

it. $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	*64* $\mapsto$ *12*	*192* $\mapsto$ *13*	*447* $\mapsto$ *84*
$S_{c,J}$	*128* $\mapsto$ *128*	*384* $\mapsto$ *383*	*893* $\mapsto$ *890*

Here we can see that $S_{c,I}$ is extremely reduced in its size while $S_{c,J}$ is only slightly changed. But for $S_{c,I}$ this is not surprising, since its condition number is very high (up to 10^9) and already the full GMRES needed almost as many iterations as the size of the system.

Again the number of iterations for a further right hand side will be reduced as side effect of the nested process.

Second right hand side of GMRES(30) combined with nested d& c

it. $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	*13*	*14*	*61*
$S_{c,J}$	*31*	*61*	*61*

For $S_{c,I}$ the number of steps for the second right hand side is extremely less than for the first right hand side, but note that the coupling system has also been extremely reduced such that we only have a small coupling system at the end. In addition the enormous reduction of $S_{c,I}$ means that multiplying with the remaining $S_{c,I}$ will be quite expensive. For $S_{c,J}$ the number of iterations for the second right hand side is only slightly less than for the first one. But here the reduction as well as the number of iterations is quite moderate and the nested divide & conquer process need not essentially reduce the coupling system $S_{c,J}$.

At last we we will show the flops *for the nested divide & conquer method for the first right hand side.*

flops *for the Solution Process of GMRES(30)*

flops $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	—	—	—
$S_{c,J}$	$1.1{\cdot}10^7$	$2.5{\cdot}10^7$	$4.1{\cdot}10^7$

flops *for the Solution Process of GMRES(30) combined with nested d& c*

flops $\backslash p$	*2*	*4*	*8*
$S_{c,I}$	$5.2{\cdot}10^7$	$2.6{\cdot}10^8$	$1.2{\cdot}10^9$
$S_{c,J}$	$1.1{\cdot}10^7$	$2.9{\cdot}10^7$	$5.1{\cdot}10^7$

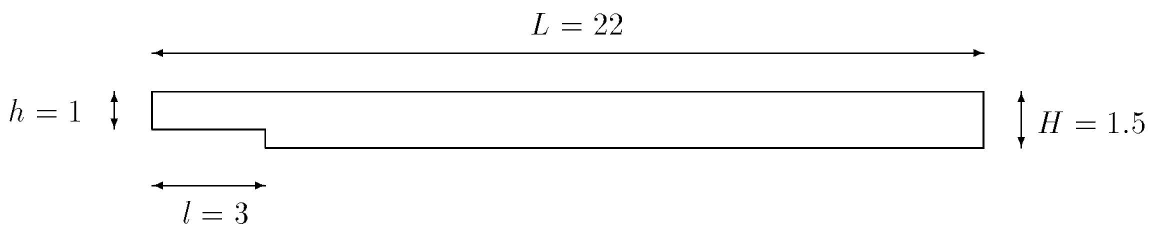
Of course the nested divide & conquer will become expensive for $S_{c,I}$ but without using the updates the process needs more than 1856 *steps and will be much more expensive. For $S_{c,J}$ the same number of flops is needed but in this case the nested divide & conquer process will only slightly change $S_{c,J}$*

At last we will examine a more realistic example from fluid dynamics.

Example 9.14 *We will consider a realistic problem.* **<u>The Generalized Stokes Problem</u>***(see e.g. [38]) in two dimensions is the following partial differential equation*

$$\begin{array}{lrcl} & -\frac{1}{Re}\Delta u + (au)_x + (bu)_y + p_x & = & 0 \\ (9.15) & -\frac{1}{Re}\Delta v + (av)_x + (bv)_y + p_y & = & 0 \\ & u_x + v_y & = & 0 \end{array}$$

where u, v, p are the desired functions depending on x and y, Re is the Reynolds number. As model domain we consider the backward facing step problem [61]. The domain can be described as follows.

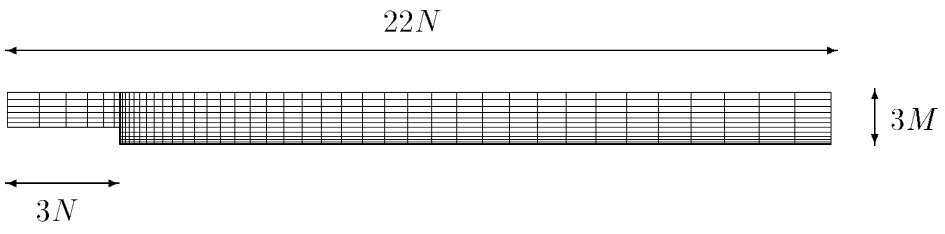


On the left end of the domain we prescribe a parabolic inflow with maximum 1 and on the right end of the domain we prescribe a parabolic outflow with maximum $2/3$.

As Reynolds numbers we will take $100, 300$ *and* 1000. *For a, b we take for simplicity*

$$a = 1, b = 0.$$

The discretization is done by the finite volume approach together with a flux–difference splitting method [22]. It has been shown in [64] that the matrices arising from this discretization are generalized M–matrices.
The domain is discretized as follows.



The numbers M, N determine the number of cells in horizontal and vertical direction and the size of the corresponding matrix. As the picture shows the grid has been graded with respect to lower left corner of the large domain. This is done in order to be closer to the physical problem. A very simple strategy was used for the cell partitioning, i.e., the size any cell is linearly increasing with the distance from lower left corner of the large domain.

The matrix, a generalized M–matrix, arising from this problem can be partitioned into blocks of size 3×3 such that any block is a symmetric 3×3 matrix. Moreover the diagonal blocks are positive definite and the off–diagonal blocks are negative definite. In order to split the initial matrix $A = S - W$ we adapt the initial modified block Jacobi splitting as follows. Without modifications W_J from the standard block Jacobi splitting would consist of matrices of the form

$$\begin{pmatrix} O & C \\ D & O \end{pmatrix},$$

where C and D are symmetric positive semidefinite matrices. In this case there exists G such that $[C, D] = [G\Lambda_C G^T, G\Lambda_D G^T]$ and Λ_C, Λ_D are positive definite diagonal matrices. As modified S we use the block diagonal matrix such that W has locally the form

$$\begin{pmatrix} G\Lambda_C^{1/2} \\ G\Lambda_D^{1/2} \end{pmatrix} \begin{pmatrix} \Lambda_D^{1/2} G^T & \Lambda_C^{1/2} G^T \end{pmatrix}.$$

The reason for this choice is that the modifications made for S give symmetric positive definite diagonal blocks and inherit the given initial symmetry structure.

We will examine this problem for the values

M	4	5	8
N	2	3	4
p	4	8	16

The size of the system will approximately increase by 2 if p increases by 2. In fact we have for the size n of the system

p	4	8	16
n	1476	2754	5868

First of all we will compare the condition numbers of the related coupling systems.

Condition Number, Re $= 100$

$\kappa_2 \backslash p$	*4*	*8*	*16*
$\kappa_2(S_{c,I})$	$1.0 \cdot 10^3$	$9.1 \cdot 10^4$	$1.3 \cdot 10^5$
$\kappa_2(S_{c,mod})$	$4.0 \cdot 10^3$	$1.4 \cdot 10^5$	$2.8 \cdot 10^6$
$\kappa_2(S_{c,fm})$	$8.2 \cdot 10^3$	$2.9 \cdot 10^5$	$2.9 \cdot 10^6$
$\kappa_2(S_{c,J})$	$1.7 \cdot 10^3$	$4.4 \cdot 10^3$	$1.3 \cdot 10^4$

Condition Number, Re $= 300$

$\kappa_2 \backslash p$	*4*	*8*	*16*
$\kappa_2(S_{c,I})$	$9.6 \cdot 10^2$	$2.4 \cdot 10^3$	$1.8 \cdot 10^4$
$\kappa_2(S_{c,mod})$	$2.8 \cdot 10^2$	$4.8 \cdot 10^3$	$5.4 \cdot 10^4$
$\kappa_2(S_{c,fm})$	$2.9 \cdot 10^2$	$6.6 \cdot 10^3$	$4.4 \cdot 10^4$
$\kappa_2(S_{c,J})$	$1.8 \cdot 10^3$	$4.1 \cdot 10^3$	$4.3 \cdot 10^2$

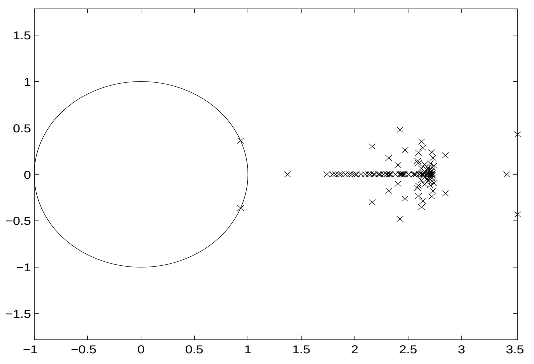
Condition Number, Re $= 1000$

$\kappa_2 \backslash p$	*4*	*8*	*16*
$\kappa_2(S_{c,I})$	$8.3 \cdot 10^2$	$3.1 \cdot 10^3$	$1.4 \cdot 10^4$
$\kappa_2(S_{c,mod})$	$2.2 \cdot 10^3$	$5.9 \cdot 10^3$	$2.8 \cdot 10^4$
$\kappa_2(S_{c,fm})$	$2.2 \cdot 10^3$	$7.0 \cdot 10^3$	$2.8 \cdot 10^4$
$\kappa_2(S_{c,J})$	$1.3 \cdot 10^3$	$4.4 \cdot 10^3$	$1.9 \cdot 10^2$

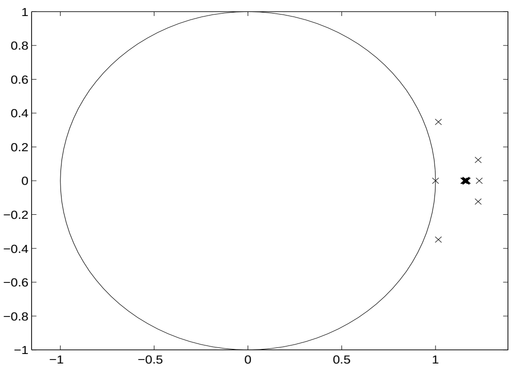
For this example the condition number of $S_{c,mod}$, $S_{c,fm}$ *was not improved when using the modified block Jacobi splitting. It was often worse then for the simple choice* $X = I$, *but not much worse. It is clear that from the construction of* X *we do not have any guarantee that we will get an improvement in the condition number.*

For the special case that $a = 1, b = 0$ *and Re* $= 100, 300, 1000$ *we will compare the eigenvalues of* $S_{c,I}$ *with those of* $S_{c,mod}$ *and* $S_{c,fm}$. *First we will consider the case Re* $= 100$ *and compare the eigenvalue distribution of* $S_{c,I}$ *with that of* $S_{c,mod}$.

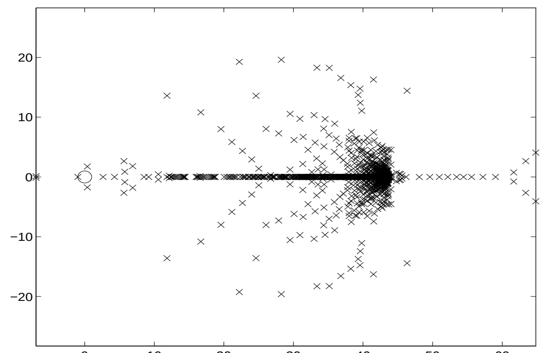
Eigenvalues of $S_{c,I} \cdot 2.7,\ p = 4$



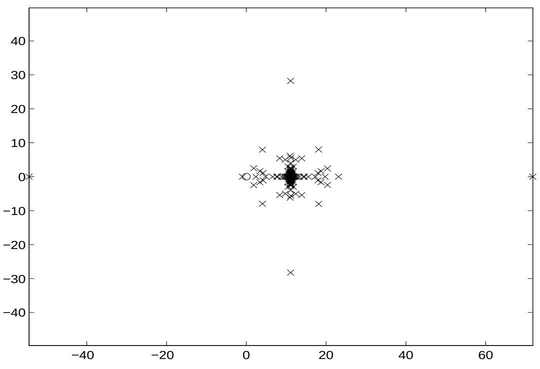
Eigenvalues of $S_{c,mod} \cdot 9.8,\ p = 4$



Eigenvalues of $S_{c,I} \cdot 4.4 \cdot 10^1,\ p = 16$

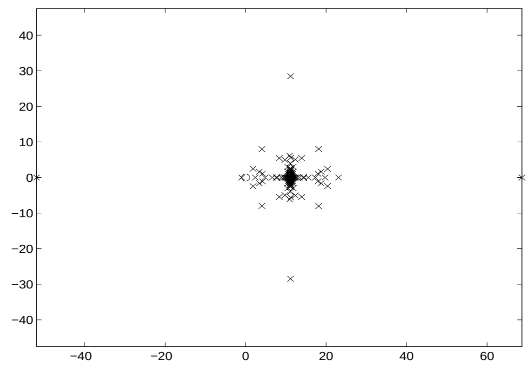


Eigenvalues of $S_{c,mod} \cdot 5.5 \cdot 10^3,\ p = 16$



For $p = 4$ *the eigenvalues of* $S_{c,mod}$ *are extremely clustered compared with those of* $S_{c,I}$. *For* $p = 16$ *the eigenvalues are still more clustered for* $S_{c,mod}$ *but additional spread eigenvalues are coming in (note that the eigenvalue plots have different scaling!). For* $S_{c,fm}$ *the eigenvalue distribution is quite analogous to that of* $S_{c,mod}$. *We illustrate this for* $p = 16$.

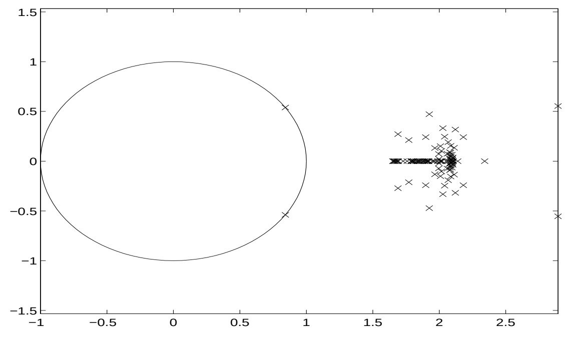
Eigenvalues of $S_{c,fm} \cdot 5.0 \cdot 10^3,\ p = 4$



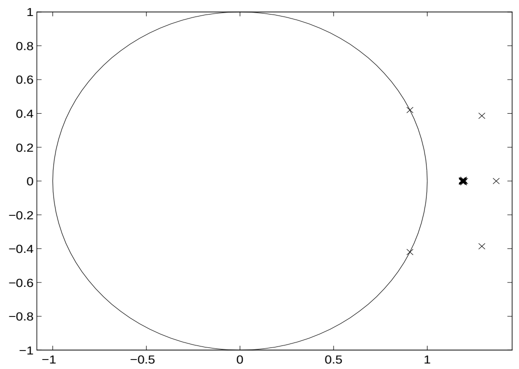
An analogous effect was observed for Re $= 300, 1000$. *For this reason we will skip the eigenvalue plots of* $S_{c,fm}$.

Next we will consider the case Re $= 300$.

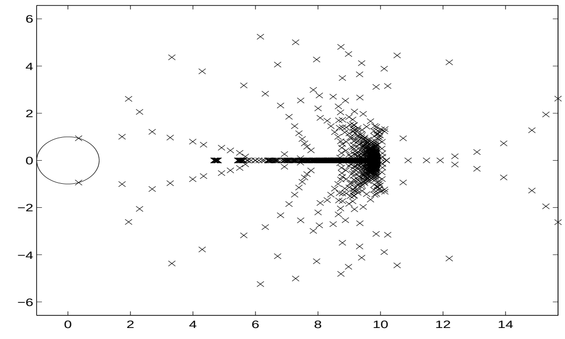
Eigenvalues of $S_{c,I} \cdot 2.1,\ p = 4$



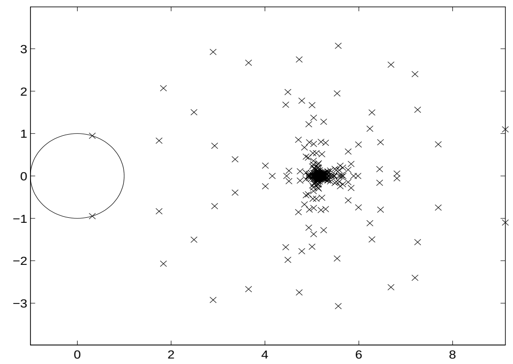
Eigenvalues of $S_{c,mod} \cdot 8.3,\ p = 4$



Eigenvalues of $S_{c,I} \cdot 9.9,\ p = 16$



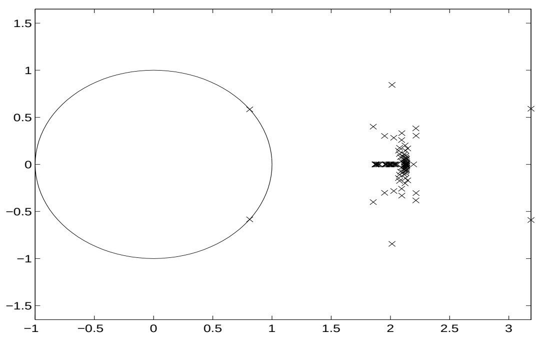
Eigenvalues of $S_{c,mod} \cdot 3.4 \cdot 10^1,\ p = 16$



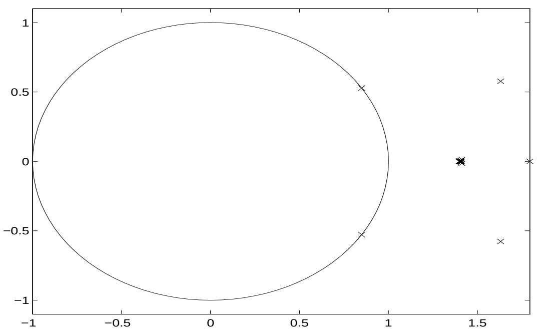
Here for $p = 4$ *the eigenvalues of* $S_{c,I}$ *are more clustered than for* $Re = 100$ *but for* $S_{c,mod}$ *the eigenvalues are even more clustered. For both matrices the eigenvalues are much more clustered when* $Re = 300$ *than for* $Re = 100$. *Here extremal spread eigenvalues like for* $Re = 100$ *do not occur for* $S_{c,mod}$. *The clustering of the eigenvalues is still better for* $S_{c,mod}$ *than for* $S_{c,I}$.

At last we will consider the case $Re = 1000$.

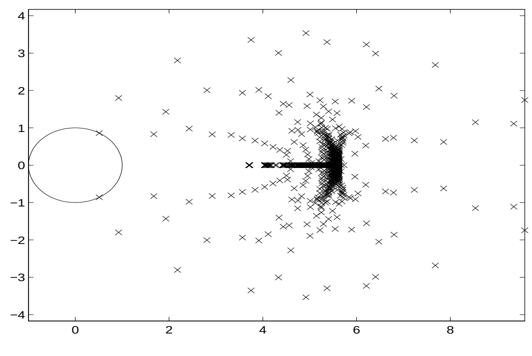
Eigenvalues of $S_{c,I} \cdot 2.1,\ p = 4$



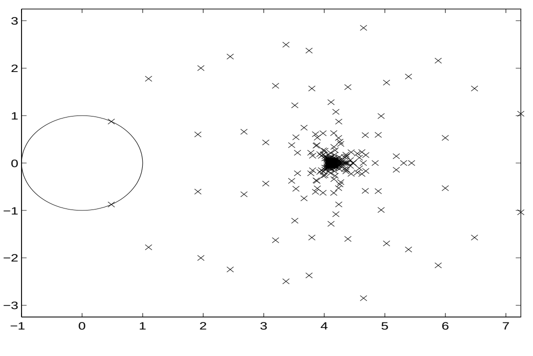
Eigenvalues of $S_{c,mod} \cdot 7.8,\ p = 4$



Eigenvalues of $S_{c,I} \cdot 5.6,\ p = 16$



Eigenvalues of $S_{c,mod} \cdot 2.4 \cdot 10^1,\ p = 16$



Here we observe that the eigenvalues of $S_{c,I}$ *and* $S_{c,mod}$ *are more clustered than for* $Re = 300$ *and in this case the improvement using* $S_{c.mod}$ *is less than for* $Re = 300$.

Beside the condition number which is unfortunately not improved for this problem we will examine the GMRES method applied to $S_J^{-1}A$, $S_{c,I}$ $S_{c,mod}$ and $S_{c,fm}$.

Number of GMRES Steps, Re $= 100$

p	4	8	16
$S_{c,I}$	20	41	90
$S_{c,mod}$	11	22	53
$S_{c,fm}$	11	22	53
$S_J^{-1}A$	16	27	51

The number of iterations for all four algorithms is quite moderate. Compared with $S_{c,I}$, we observe that $S_{c,mod}$,$S_{c,fm}$ will need much less iteration steps. The situation for all algorithms will become better when increasing the Reynolds number.

Number of GMRES Steps, Re $= 300$

p	4	8	16
$S_{c,I}$	19	33	65
$S_{c,mod}$	11	22	46
$S_{c,fm}$	11	22	46
$S_J^{-1}A$	15	27	47

Number of GMRES Steps, Re $= 1000$

p	4	8	16
$S_{c,I}$	20	33	57
$S_{c,mod}$	12	22	45
$S_{c,fm}$	12	23	45
$S_J^{-1}A$	15	26	46

The improvement made by using the modified block Jacobi–Splitting is quite small since the number of iterations for all methods is very moderate.

Note that here we used full GMRES. In practice one typically uses a restarted GMRES, i.e. after k steps of the iteration one stops and begins the iteration again. Now we will examine for $p = 16$ *how the number of iterations will change when using the GMRES method with 30 steps. In addition we will combine the restarted GMRES method with the nested divide & conquer method. In order to be able to compare the GMRES(30) with and without using low rank updates we will apply for the block Jacobi method the GMRES iteration to the related coupling system $S_{c,J}$. Since $S_{c,fm}$ and $S_{c,mod}$ show a similar behaviour we will not examine $S_{c,mod}$ in the next computations.*

First we will compare the number of iterations when not using the nested divide & conquer.

Number of Iteration Steps of GMRES(30), $p = 16$

it. \ *Re*	*100*	*300*	*1000*
$S_{c,I}$	$> n$	*121*	*91*
$S_{c,fm}$	*241*	*61*	*61*
$S_{c,J}$	*121*	*91*	*61*

When replacing the full GMRES by the GMRES(30) method for the coupling system the number of iterations increases drastically particularly for Re $= 100$. *For Re* $= 300, 1000$ *the situation is better. For* $S_{c,I}$ *we do not have convergence after* n *steps. The situation is better for* $S_{c,fm}, S_{c,J}$ *but still unsatisfactory.*

The situation changes in combination with the nested divide & conquer method. Here we need the following number of steps.

Number of Iteration Steps of GMRES(30) combined with nested d& c, $p = 16$

it. \ *Re*	*100*	*300*	*1000*
$S_{c,I}$	*181*	*91*	*60*
$S_{c,fm}$	*118*	*59*	*53*
$S_{c,J}$	*91*	*90*	*61*

Here the number of iterations for Re $= 100$ *has been significantly reduced. For Re* $= 300, 1000$ *the difference is less strong but here the GMRES(30) is much more moderate. For* $S_{c,I}$ *and Re* $= 100$ *we did not have convergence after* n *steps but this has drastically changed now.*

During the nested divide & conquer process the coupling system has been successively reduced in its size.

Size of the coupling system, initially $\mapsto$ *finally*

it. \ *Re*	$\mapsto$	*100*	*300*	*1000*
$S_{c,I}$	*1041* $\mapsto$	*1019*	*1035*	*1027*
$S_{c,fm}$	*1041* $\mapsto$	*1026*	*1035*	*1031*
$S_{c,J}$	*2082* $\mapsto$	*2066*	*2072*	*2078*

For $S_{c,I}$, *Re* $= 100$ *the reduction is bigger than in all other cases but here the problem of convergence was most critical without using the nested divide & conquer. For the other cases only a small reduction has been done and except for the case Re* $= 100$ *for* $S_{c,fm}$ *the reduction in the number of iterations is small as well.*

Finally we will compare the flops *when using the nested divide & conquer for one right hand side.*

flops *for the Solution Process of GMRES(30)*

flops \ *Re*	*100*	*300*	*1000*
$S_{c,I}$	—	$1.7\cdot10^8$	$1.3\cdot10^8$
$S_{c,fm}$	$2.1\cdot10^8$	$5.0\cdot10^7$	$5.0\cdot10^7$
$S_{c,J}$	$1.3\cdot10^8$	$9.6\cdot10^7$	$6.3\cdot10^7$

flops *for the Solution Process of GMRES(30) combined with nested d& c,* $p = 16$

flops \ *Re*	*100*	*300*	*1000*
$S_{c,I}$	$3.8\cdot10^8$	$1.6\cdot10^8$	$1.2\cdot10^8$
$S_{c,fm}$	$1.4\cdot10^8$	$6.2\cdot10^7$	$6.3\cdot10^7$
$S_{c,J}$	$1.6\cdot10^8$	$1.4\cdot10^8$	$7.8\cdot10^7$

For those cases where the GMRES(30) without using the nested divide & conquer method has lead to a larger number of iterations the nested divide & conquer could effectively reduce the number of iterations as well as the computational time. For those cases where the number of iteration steps was already quite moderate without using the nested divide & conquer the number of iterations was only slightly reduced but the computational costs even increase slightly. The problem for the heuristic that we have used is that it does not give a guarantee that the number of iterations will be reduced and the additional computational work can not always equalize the additional work. But more important is the fact that for large number of iterations we could reduce the number of iterations as well as the computational costs and in addition the size of the system that has to be solved has been reduced.

Summary

We have illustrated for some examples the use of modified block Jacobi splittings as well as the application of the nested divide & conquer method. Modified block Jacobi splittings in general were used to improve the properties of the coupling system. In Example 9.12 and 9.13 an improvement in the condition number and the eigenvalue distribution has been observed. However in Example 9.14 there was no improvement in the condition number. The condition number of the coupling system using the block diagonal part of the approximate solution X of the algebraic Riccati equation from Chapter 6 has partially lead to worse condition number. The problem in general is that choosing the block diagonal part of X will be most likely not the best choice and so far there is no theory which block diagonal approximation of X should be chosen. Thus an improvement of the condition number cannot be expected in general. But what has been observed is that the eigenvalue clustering of the coupling system has been improved. In Example 9.14 this effect is not as strong as in the other examples. One problem in general concerning modified block Jacobi splittings is that we have required the nonsingularity of the block diagonal part of the initial matrix. This currently reduces the applicability of modified block Jacobi splittings and future investigations have to discuss the case when the block diagonal part of A is singular. A general problem in the practical realization of modified block Jacobi splittings is that

the computation of the approximate solution X of the algebraic Riccati equation requires the solution of several right hand sides with a block diagonal matrix. If e.g. A is block tridiagonal with all blocks of size $m \times m$, then $2m$ systems with a block diagonal matrix have to be solved. This make the generation of X expensive if an iterative method is used and if this iterative method only needs a moderate number of steps, i.e. less than $2m$ steps. Of course in general the number of iterations may become arbitrary many. In this case we can exploit the fact that computing X will also give us the opportunity to have an explicit representation of the coupling system. For a direct solution as well as for an iterative solution of the coupling system this will essentially reduce the costs of the solution process. For an iterative solution much of the time consumed by the generation of X can be equalized by the cheaper matrix vector multiplication.
Here we have only compared the sequential flop count. When being implemented on a parallel machine one has to take into account that additional communication has to be done. Here the generation of $\bar{D}, D$ has the advantage, that the their computation can be done without communication. This aspect is analogous to the symmetric case, where we have already discussed the problem.

Modified block Jacobi splittings as well as the unmodified block Jacobi splitting form the basis for the nested divide & conquer process. If the (modified) splitting exists then the nested divide & conquer method is applicable. The nested divide & conquer process has been applied to all examples using a simple heuristic for constructing updates. This strategy here has been constructed using the GMRES method and the eigenvalue information which is produced by this algorithm. The strategy essentially tries to find spread eigenvalues and eigenvalues of small modulus. In contrast to other strategies like [73],[53],[62] here we have used the corresponding approximate invariant subspace to reduce the size of the coupling system. Like other strategies there is no guarantee that the iterative process will converge faster when using this update strategy. But here we have always reduced the size of the coupling system which will finally lead to direct method even if the iterative process would fail. This semi–direct approach makes the nested divide & conquer process applicable to a large class of problems. The problem which is open in general is how the updates should be chosen. To use this simple heuristical strategy by detecting spread eigenvalues and eigenvalues of small modulus will be most likely not the best strategy. But in any case this strategy will tend more and more to a direct method the longer the iteration will take.

Conclusions

We have developped an algebraic strategy of domain decomposition for large sparse linear systems, which is based on the low rank modification formula of Sherman, Morrison and Woodbury. The advantage of this approach is its high flexibility, since only low rank splittings of the form $A = S - W$ with W of small rank are required. From the point of view of parallel computations we need to specify the low rank splitting and here we have used block diagonal splittings. In general we need a block diagonal splitting with a nonsingular block diagonal matrix S such that S and much more the related coupling system is not ill–conditioned.

To get a block diagonal matrix an algorithm that partitions the matrix with respect to the underlying graph has to be used. In principle this algorithm has to partition the graph such that as few edges as possible have to be taken out to get the block diagonal part of A. If a block diagonal matrix is obtained by this partitioning, then we can modify the block diagonal matrix with respect to certain aspects. For certain classes of matrices, symmetric positive definite matrices, M–matrices we can modify the block diagonal part in order to inherit structures for the block diagonal part S as well as for the coupling system. Beside this modifications we can locally minimize the rank of the remaining matrix. Here there are still restrictions in the general case. So far we still need the nonsingularity of the block diagonal part of A, i.e. we need that it is not ill–conditioned. In addition we have as requirement, that the block graph has to be block 2–cyclic. The modifications then can be traced back to the solution of an algebraic Riccati–equation and under relatively general assumptions explicit solution exist. For practical purposes and the application to the initial block diagonal matrix these solutions have to be approximated by a matrix which is itself block diagonal. So far it is open which block diagonal matrix has to be taken and taking the block diagonal part of the solution is most likely not the best choice in the general case. In contrast to this in the symmetric positive definite case we have derived bounds on the optimality of this modifications.
Based on the given splitting we apply the Sherman–Morrison–Woodbury formula to solve the problem in parallel. The main problem is to solve the coupling system of small size which is involved by this formula. To make a compromise between a direct and an iterative solution of the coupling system, the Sherman–Morrison–Woodbury formula has been successively applied in order to reduce the size of the coupling system and thus to reduce the rank of the remaining matrix. By this nested application the initial splitting has been adaptively modified by low rank updates. The nested application of the Sherman–Morrison–Woodbury formula can be read as performing an LU decomposition of a suitably extended system and likewise the remaining coupling system which is generated by the nested divide & conquer strategy can be obtained from an LU decomposition of the initial coupling system after a suitable pre– and post multiplication. From this observation close connections to algebraic multigrid methods could be derived and results have been transferred to the nested divide & conquer strategy. Here we have used orthogonal transformations and for the symmetric positive definite case optimal transformations in the sense of quadratic forms have been derived. In the general case it is still open which kind of transformation has to be chosen. In any case a reduction of the coupling system

can be achieved and even if an iterative method would fail we reduce the coupling system by this approach.
For this algebraic concept a parallel realization has been developped. Here the block graph of the block diagonal splitting induces a natural distribution of initial system and also for the related coupling system. We have transferred the use of so–called adding type vectors and overlapping type vectors to our algebraic problem which has lead to nice treatment of the coupling system in parallel. To generalize this parallel realization to the nested use of the Sherman–Morrison–Woodbury formula, we have presented a strategy that collects the low rank updates to one matrix in order to keep the data traffic small. Here the fact that implicitly an LU decomposition of the coupling system is made has been turned out to be quite useful.
In the numerical experiments we have compared the modified block diagonal splittings with the unmodified splittings. While in the positive definite case an effective improvement of the condition number has been illustrated, this is not necessarily true in the general case. What has been observed in the general case is an improvement in the eigenvalue distribution. Here the gap in the choice of the block diagonal matrix from the Riccati–equation has to be closed. We have observed that in several cases the iterative process (CG,GMRES) will be accelerated by this choice but there also exist counter examples. For the application of the nested divide & conquer strategy we have used a heuristical approach in the unsymmetric case to define an orthogonal transformation which is based on the eigenvalue information of the underlying Arnoldi process in the GMRES method. By choosing updates with respect to this heuristic we cannot necessarily reduce the number of iterations in general, but in any case the size of the coupling system will be reduced and finally the nested divide & conquer strategy becomes more and more a direct method. In the symmetric positive definite case we have illustrated how the nested divide & conquer process can be used on a parallel computer and in the numerical examples the nested divide & conquer strategy has turned out as compromise between a direct and iterative solution that can overcome problems with the iterative process especially when the system is ill–conditioned and the iterative process will extremely slow down or does not converge at all. An additional nice aspect in using the nested divide & conquer process is that solving further right hand sides will be much cheaper than the first right hand side.

What we cannot handle by this algebraic concept so far is the application of general matrices with possibly singular block diagonal part. The problem that has to be solved is that the block diagonal matrix S must be nonsingular and moreover it must not be too ill–conditioned. At the same time we need that the remaining part $W = S - A$ has low rank. Adaptions in this direction still have to be done. By the nested divide & conquer method we cannot ensure scalability, since on one hand information from the underlying eigenvalue process is needed and on the other hand too many low rank updates may still slow down this approach.

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen: LAPACK Users' Guide, Second Edition. SIAM Philadelphia, 1995.

[2] O. Axelsson: A Survey of Preconditioned Iterative Methods for Linear Systems of Algebraic Equations. BIT, **25** (1985), pp.166–187.

[3] O. Axelsson: Iterative Solution Methods. Cambridge University Press, 1993.

[4] O. Axelsson, G. Lindskog: On the Eigenvalue Distribution of a Class of Preconditioned Methods. Numer. Math., **48** (1986),pp.479–498.

[5] O. Axelsson, G. Lindskog: On the Rate of Convergence of the Preconditioned Conjugated Gradients Method. Numer. Math., **48** (1986),pp.499–523.

[6] P. Bastian: Parallele adaptive Mehrgitterverfahren. B.G. Teubner Stuttgart, 1996.

[7] L. Becks, D. Rogowski: Konvergenzanalyse und Implementierung eines algebraischen Mehrgitterverfahrens zur Lösung linearer Gleichungssysteme. Diplomarbeit, Universität Bielefeld, 1990.

[8] A. Berman, R.J. Plemmons: Nonnegative Matrices in the Mathematical Sciences. Classics in Applied Mathematics, SIAM, Philadelphia, 1994.

[9] P.E. Bjørstad, O.B. Widlund: Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures. SIAM J. Numer. Anal., **23** (1986), pp.1097–1120.

[10] M. Bollhöfer: Algebraic Domain Decomposition. Preprint SPC 95_11, TU Chemnitz-Zwickau, March 1995.

[11] M. Bollhöfer, C. He, and V. Mehrmann: Modified Block Jacobi Preconditioners for the Conjugate Gradient Method. Part I: The Positive Definite Case. Preprint SPC 95_7, TU Chemnitz-Zwickau, January 1995.

[12] S. Bondeli: Parallele Algorithmen zur Lösung tridiagonaler Gleichungssysteme. Dissertationsschrift, ETH Zürich, 1991.

[13] J. A. Bondy, and U. S. R. Murty: Graph Theory with Applications. The MacMillan Press Ltd, 1976.

[14] D. Braess: Finite Elemente. Springer-Verlag, 1992.

[15] J.H. Bramble, J.E. Pasciak, A.H. Schatz: The Construction of Preconditioners for Elliptic Problems by Substructuring I. Math. Comp., **47** (1986), pp.103–134.

[16] A. Brandt: Algebraic Multigrid Theory: the Symmetric Case. Appl. Math. Comp., **19** (1986), pp.23–65.

[17] C.W. Brand, S. Petrova: Preconditioned Iterations to Calculate Extreme Eigenvalues. Technical Report, Montanuniversiät Leoben, Inst. f. Angew. Mathem., 1993.

[18] W. Bunse, A. Bunse–Gerstner: Numerische lineare Algebra. Teubner, Stuttgart, 1985.

[19] D. Calvetti, L. Reichel, and D.C. Sorensen. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. Electr. trans. Num. Anal., **2** (1994), pp. 1–21.

[20] W. Dahmen, L. Elsner: Hierarchical Iteration. Lecture Notes on Numerical Fluid Dynamics, Hackbusch (ed.), Vieweg, 1988.

[21] J. Demmel: The condition number of equivalence transformations that block diagonalize matrix pencils. In B. Kagström and A. Ruhe, eds., Matrix Pencils, pp. 2-16, Berlin, FRG, 1982. Springer Verlag. Lecture Notes in Mathematics 973.

[22] E. Dick, J. Linden: A multigrid flux–difference splitting method for steady incompressible Navier–Stokes equations. Proc. of the GAMM Conference on Numerical Methods in Fluid Mechanics, Delft, 1989.

[23] J.J. Dongarra, C.B. Moler, J.R. Bunch, G.W.Stewart: LINPACK Users' Guide. SIAM Philadelphia, 1979.

[24] M. Dryja: A Capacitance Matrix Method for Dirichlet Problem on Polygonial Region. Numer. Math., **39** (1982), pp.51–64.

[25] M. Dryja, O.B. Widlund: Some Domain Decomposition Algorithms for Elliptic Problems. Iterative Methods for Large Linear Systems, L. Hayes, D. Kincaid (eds.), Academic Press, Orlando, Florida, 1989.

[26] M. Dryja, O.B. Widlund: Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems. Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Périaux, O. Widlund (eds.), SIAM, Philadelphia, 1990.

[27] I.S. Duff, A.M. Erisman, J.K. Reid: Direct Methods for Sparse Matrices. Oxford University Press, 1986.

[28] I.S. Duff, R.G. Grimes, and J.G. Lewis: Sparse matrix test problems. ACM Trans. Math. Software **15** (1989), pp. 1–14.

[29] M. Eiermann: Semi–Iterative Methods. Proceeding of the LSSC-course, FSP–Mathematisierung, Universität Bielefeld, 1992.

[30] S.C. Eisenstat, J.W. Lewis, and J.W. Schultz: Optimal block diagonal scaling of block 2–cyclic matrices. Lin. Alg. Appl. **44** (1982), pp. 181–186.

[31] L. Elsner, V. Mehrmann: Convergence of block iterative methods for linear systems arising in the numerical treatment of Euler equations. Numerische Mathematik **59** (1991), pp.541–559.

[32] L. Elsner, C. He, and V. Mehrmann: Minimization the Condition Number of a Positive Definite Matrix by Completion. Numer. Math. **69** (1994), pp.17–24.

[33] L. Elsner, C. He, and V. Mehrmann: Minimization of the Norm, the Norm of the Inverse and the Condition Number of a Matrix by Completion. Numer. Lin. Alg. w. Appl. Vol. **2(2)**, (1995), pp.155–171.

[34] J. Falkner, F. Rendl, H. Wolkowicz: A Computational Study of Graph Partitioning. Technical Report, University of Waterloo, 1993.

[35] D.G. Feingold, R.S. Varga: Block diagonally dominant matrices and generalization of the Gershgorin circle theorem. Pacific J. Math., **12** (1962), pp.1241–1250.

[36] M. Fiedler: Algebraic Connectivity of Graphs. Czech. Math. J. **23** (1973), pp. 298–305.

[37] R.W. Freund, G.H. Golub, N.M. Nachtigal: Iterative Solution of Linear Systems. Acta Numerica (1992), pp. 1–44.

[38] C. Frohn-Schauf: Flux-Splitting Methoden und Mehrgitterverfahren für hyperbolische Systeme mit Beispielen aus der Strömungsmechanik. Dissertation, Universität Düsseldorf, 1992.

[39] J. Fuhrmann: Zur Verwendung von Mehrgitterverfahren bei der numerischen Behandlung elliptischer partieller Differentialgleichungen zweiter Ordnung mit variablen Koeffizienten. Dissertationsschrift, TU Chemnitz–Zwickau, 1994.

[40] A. George, J. Liu: Computer Solution of Sparse Positive Definite Systems. Prentice–Hall, Englewood Cliffs, New Jersey, 1981.

[41] G.H. Golub, C.F. Van Loan: Matrix Computations, second. ed. The Johns Hopkins University Press, 1989.

[42] A. Greenbaum, V. Ptak, and Z. Strakos: Any Nonincreasing Convergence Curve is Possible for GMRES. SIAM J. Matrix Anal. Appl., **17** (1996), pp. 465-469.

[43] M. Griebel: Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen. Teubner Stuttgart, 1994.

[44] G. Haase, T. Hommel, A. Meyer, and M. Pester: Bibliotheken zur Entwicklung paralleler Algorithmen. Preprint SPC 95_20, TU Chemnitz-Zwickau, March 1995.

[45] G. Haase, U. Langer, and A. Meyer: The Approximate Dirichlet Domain Decomposition Method. Part I: An Algebraic Approach. Computing **47** (1991), pp.137–151.

[46] G. Haase, U. Langer, and A. Meyer: The Approximate Dirichlet Domain Decomposition Method. Part II: Application to 2nd–order Elliptic B.V.P.s. Computing **47** (1991), pp.153–167.

[47] W. Hackbusch: Multigrid Methods and Applications. Springer, Berlin,Heidelberg, 1985.

[48] W. Hackbusch: Theorie und Numerik elliptischer Differentialgleichungen. Teubner, Stuttgart, 1986.

[49] W. Hackbusch: Iterative Lösung großer schwachbesetzter Gleichungssysteme. Teubner, Stuttgart, 1991.

[50] M. Hochbruck, C. Lubich: Error Analysis of Krylov Methods in a Nutshell. Preprint Mathematisches Institut, Universität Tübingen, August 1995, revised Feb 1996, to appear in SIAM J. Sci. Comput.

[51] W. Kahan, and B.N. Parlett: An Analysis of Lanczos Algorithms for Symmetric Matrices. ERL-M467, University of Califronia, Berkely, 1974.

[52] G. Karypis, V. Kumar: A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95–035. Dept. of Computer Science, University of Minnesota, 1995.

[53] S.A. Kharchenko, and A. Yu. Yeremin: Eigenvalue Translation Based Preconditioners for the GMRES(k) Method. Numerical Linear Algebra with Appilcations, Vol. **2(1)**,(1995), pp.51–77.

[54] P. Lancaster, L. Rodman: The Algebraic Riccati Equation. Oxford University Press, Oxford, 1995.

[55] K.H. Law: A parallel finite element solution method. Computer and Structures, **23(6)**, (1989), pp.845–858.

[56] R.B. Lehoucq: Analysis and Implementation of an Implicitly Restarted Iteration. PhD thesis, Rice University, Houston, Texas, May 1995.

[57] L. Mansfield: On the Conjugate Gradient Solution of the Schur–Complement System Obtained from Domain Decomposition. SIAM J. Numer. Anal., **27(6)**,(1990), pp.1612–1620.

[58] V. Mehrmann: The Autonomous Linear Quadratic Control Problem. Springer, 1991.

[59] V. Mehrmann: Divide & Conquer Methods for Block Tridiagonal Systems. Parallel Computing **19** (1993), pp. 257–279.

[60] C. Moler: MATLAB User's Guide. Technical Report CS81-1, Dept. of Computer Science, University of New Mexico, Albuquerque, 1980.

[61] K. Morgan, J. Periaux, and F. Thomas: Analysis of Laminar Flow over a Backward Facing Step. Notes on Numerical Fluid Mechanincs, Vol.9, Vieweg Verlag, Braunschweig, 1984.

[62] R. B. Morgan: A Restarted GMRES Method Augmented With Eigenvectors. SIAM J. Matrix Anal. Appl. **16(4)**,(1995), pp. 1154–1171.

[63] R. Nabben: On a new class of matrices which arise in the numerical solution of Euler equations. Num. Math., **63** (1992), pp. 411–431 .

[64] R. Nabben: A new application for generalized M–matrices. Numerical Linear Algebra (Kent, OH, 1992), pp.179–192, de Gruyter, Berlin, 1993.

[65] S.V. Nepomnyashchikh: Domain Decomposition and Schwarz Methods in a Subspace for the Approximate Solution of Elliptic Boundary Value Problems (russ.). PhD thesis, Novosibirsk 1986.

[66] E.E. Osborne: On Pre–Conditioning of Matrices. Jour. ACM **7** (1960), pp. 338–345.

[67] B. N. Parlett: The Symmetric Eigenvalue Problem. Prentice-Hall, Englewood Cliffs, N.J., 1980.

[68] B.N. Parlett and, C. Reinsch: Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors. Numer. Math. **13** (1969), pp.293–304.

[69] B. N. Parlett, and D.S. Scott: The Lanczos Algorithm with Selective Reorthogonalization. Math Comp. **33**(1979), pp.217-238.

[70] F. Rendl, H. Wolkowicz: A Projection Technique for Partitioning the Nodes of a Graph. Technical Report, University of Waterloo, 1990.

[71] M. Rozlǒzník: Numerical Stability of the GMRES Method. PhD Thesis, Institute of Computer Science, Acadamy of Sciences of the Czech Republic, April 1997.

[72] J.W. Ruge, K. Stüben: Algebraic Multigrid. Multigrid Methods, McCormick (ed.), SIAM Philadelphia, 1987.

[73] Y. Saad: Projection and deflation methods for partial pole assignment in linear state feedback. IEEE Trans. Automat. Contr. **33(3)**, (1988), pp.290-297.

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

[75] Y. Saad, and M. Schultz: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Computing **7**(1986), pp.856–869.

[76] R. Schreiber, B.N. Parlett: Block reflectors: Theory and Computation. SIAM J. Numer. Anal. **25**(1987), pp.189–205.

[77] R. Schreiber, C.F. Van Loan: A Storage Efficient WY Representation for Products of Householder Transformations. SIAM J. Sci and Stat. Comp. **10** (1989), pp. 52–57.

[78] H.A. Schwarz: Gesammelte Mathematische Abhandlungen. Vierteljahresschrift der Naturforschenden Gesellschaft in Zürich, **15** (1870),pp.272–286.

[79] H.R. Schwarz: Methode der finiten Elemente. Teubner, Stuttgart, 1991.

[80] H.D. Simon: Partitioning of Unstructured Problems for Parallel Processing. Computing Systems in Engineering, **2** (1991), pp.135–148.

[81] D.C. Sorensen: Implicit application of polynomial filters in a k–step Arnoldi method. SIAM J. Matrix Anal. Appl., **13** (1992), pp.357–385.

[82] R.S. Varga: Matrix Iterative Analysis. Prentice Hall, Englewood–Cliffs, N.J., 1962.

[83] O.B. Widlund: Iterative Substructuring Methods: Algorithms and Theory for Elliptic Problems in the Plane. First International Symposium on Domain Decomposition Methods for Partial Differential Equations. R. Glowinski, G. Golub, G. Meurant, J. Périaux (eds.), SIAM, Philadelphia, 1988.

[84] O.B. Widlund: Optimal Iterative Refinement Methods. Domain Decomposition Methods, Proceedings Jan. 1988, Los Angeles. Chan–Glowinsky–Periaux–Widlund (ed.), SIAM, Philadelphia 1989, pp.114–125.

[85] J. Xu: Iterative Methods by Space Decomposition and Subspace Correction. SIAM Review **34** (1992), pp.581–613.

Summary of the Thesis

We have developped an algebraic strategy of domain decomposition for large sparse linear systems, which is based on the low rank modification formula of Sherman, Morrison and Woodbury. The algebraic concept is based on three columns. First the nested use of the Sherman–Morrison–Woodbury formula which is applicable to any low rank splitting. The second one is the use of modified block Jacobi splittings and the factorization of the low rank part. And finally the third column is the parallel realization of the nested divide & conquer method applied to a block diagonal splitting.

As first column we have presented a nested divide & conquer strategy which consists of successively replacing the initial matrix S by S plus an additional low rank modification. By this strategy we have adaptively constructed a preconditioner of the form S plus low rank which ensures that even if an iterative method for S_c would fail we would reduce the coupling system in its size and in this case we finally end up in a direct method. By Theorem 4.23 the nested application of the Sherman–Morrison–Woodbury formula can be interpreted as implicitly performing an LU–decomposition of a suitably extended system and likewise we have an LU decomposition of S_c after a suitable a priori transformation. Optimal orthogonal transformations have been derived in Theorem 3.31 for the symmetric positive case. We have demonstrated close connections to algebraic multigrid methods. In Corollary 4.30 results for the symmetric positive definite case which have been designed for substructuring methods, have been applied for algebraic domain decomposition methods.

Especially block diagonal matrices were of interest due to their easy realization on parallel architectures. We have discussed a way to factorize the low rank part of the block diagonal splitting and for some classes of matrices we have discussed structure preserving modifications of the block diagonal part. More complicated modifications that can be read as some kind of algebraic boundary conditions have been introduced. We have shown that this problem can be traced back to finding approximate solutions of algebraic Riccati equations. For the general case there are still many questions open concerning the choice of algebraic boundary conditions while in the symmetric case we have discussed in Theorem 6.76 the optimality of the choice of algebraic boundary conditions in the sense of quadratic forms.

The third column of algebraic domain decomposition concept is its parallel realization. So–called adding type vectors and overlapping type vectors which are well–known in domain decomposition methods have been transferred to this algebraic method of domain decomposition in Theorem 7.37. To adapt the parallel treatment to the use of the nested Sherman–Morrison–Woodbury formula we have presented a concept that accumulates the low rank modifications to one block of data in order to reduce the data traffic.

The theory has been illustrated in several examples to confirm with the theoretical results and how especially a parallel realization behaves.

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Dissertation selbständig verfaßt habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden sind.

Chemnitz, den 26.11.1997, **Matthias Bollhöfer**

Thesen

zur Dissertation

Algebraic Domain Decomposition

zur Erlangung des akademischen Grades eines “Dr. rer. nat.”
an der Technischen Universität Chemnitz
Fakultät für Mathematik

vorgelegt von **Dipl.–Math. Matthias Bollhöfer**

1. Mit der Entwicklung von Parallelrechnern sind in den letzten Jahren eine Reihe von Techniken und Ansätze [5] entwickelt worden, um große Systeme $Ax = b$ mit schwachbesetzter regulärer $n \times n$ Matrix A auf Parallelrechnern numerisch zu lösen. Ein Ansatz beruht auf der Verwendung der Sherman–Morrison–Woodbury Formel zur Invertierung von Matrizen bei Modifikationen von niedrigem Rang. Ist etwa $A = S - W = S - FG$, S regulär, $W = FG$ von niedrigem Rang, so gilt

   $$(S - FG)^{-1} = S^{-1} + S^{-1}F(\underbrace{I - GS^{-1}F}_{S_c})^{-1}GS^{-1}. \qquad (9.16)$$

   Bei der parallelen Behandlung kann insbesondere S der blockdiagonale Anteil oder ein leicht modifizierter blockdiagonaler Anteil von A sein, gegebenfalls nach vorheriger Neuanordnung der Koeffizienten von A. Bei großen schwachbesetzten Systemen hat die Restmatrix W dann niedrigen Rang. Parallele Verfahren dieser Art sind z.B. in [1],[9] für blocktridiagonale Matrizen diskutiert worden.

2. Das Hauptproblem bei (9.16) ist die Behandlung des Kopplungssystems S_c auf einem Parallelrechner. Hier tritt das Problem auf, daß S_c zunächst nur implizit gegeben ist und die Verteilung des Kopplungssystems über die Prozessoren die Behandlung dieses Systems erschwert. Eine Möglichkeit wäre natürlich dieses System direkt zu lösen [1],[9]. Dieses ist, bedingt durch die Verteilung des Systems über die Prozessoren, jedoch ein sequentieller Prozess. Alternativ wäre eine iterative Lösung möglich, wie etwa [6], pp.516ff, [11], [4]. Problem bei Iterationsverfahren ist jedoch die Abhängigkeit der Iterationszahl von der speziellen Matrix [6], p.523, [3], [12]. In der Regel läßt sich dies durch den Einsatz von Vorkonditionierern [10] abmildern, das grundsätzliche Problem aber bleibt. Die parallele Verteilung des Kopplungssystems reduziert jedoch die Möglichkeiten des effizienten Einsatzes von Vorkonditionierern.

3. In der vorliegenden Dissertation wird zur numerischen Lösung großer schwachbesetzter linearer Gleichungssysteme auf Parallelrechnern ein Konzept der algebraischen Gebietszerlegung diskutiert, welches im Grundsatz auf der Verwendung der Formel (9.16) basiert. Ähnlich den numerischen Methoden bei der Gebietszerlegung in der Behandlung partieller Differentialgleichungen wird die Systemmatrix über die Prozessoren mit lokalem Speicher verteilt und der Datenaustausch findet über ein

zugehöriges Kommunikationsnetzwerk statt. Für (9.16) werden blockdiagonale Matrizen und modifizierte blockdiagonale Matrizen behandelt. Zur Lösung des Kopplungssystems S_c wird mit der geschachtelten Verwendung von (9.16) ein Kompromiß zwischen einem direkten und iterativen Verfahren behandelt. Techniken wie etwa die Verwendung addierender und überlappender Vektoren [8],[7], die unter anderem bei Gebietszerlegungsmethoden zur parallelen numerischen Behandlung partieller Differentialgleichungen zum Einsatz kommen, können hierbei auf den algebraischen Fall übertragen werden.

4. Im ersten Teil der Arbeit wird mit der geschachtelten Verwendung der Formel (9.16) ein Ansatz behandelt, der einen Kompromiß zwischen direkter und iterativer Lösung von S_c darstellt. Diese geschachtelte Divide & Conquer Strategie erlaubt es einerseits, den Rang der Restmatrix W zu reduzieren. Andererseits führt das sukzessive Ersetzen der Matrix S durch eine Matrix $\tilde{S}$, welche sich von S nur durch eine Matrix von niedrigem Rang unterscheidet, adaptiv zu einer Vorkonditionierungsmatrix für das Ausgangssystem. Hierdurch wird sukzessiv ein neues Kopplungssystem erzeugt, welches in seiner Dimension entsprechend reduziert worden ist. Dieser Ansatz konnte implizit auf eine LU–Zerlegung eines geeignet erweiterten Systems zurückgeführt worden. Entsprechend ist das aus der Divide & Conquer Strategie entstehende neue Kopplungssystem das Schur–Komplement des ursprünglichen Kopplungssystems S_c nach einer zuvor durchgeführten Vorabtransformation.

5. Zur geschachtelten Anwendung von (9.16) läßt sich die Reduktion des Kopplungssystems und damit die adaptive Anpassung der Matrix S im Prinzip auf das jeweilige Iterationsverfahren abstimmen. Für den Fall symmetrisch positiv definiter Matrizen konnten optimale orthogonale Transformationen im Sinne quadratischer Formen hergeleitet werden. Im allgemeinen Fall ist die richtige Wahl der orthogonalen Tranformation noch offen. Hier muß man sich bisher mit Heuristiken begnügen. Da hier gleichzeitig die Dimension der Restmatrix reduziert wird, hat man aber auch für den Fall, daß das iterative Verfahren versagt, das System in seiner Dimension sukzessive reduziert und erhält dann ein direktes Verfahren.

6. Die geschachtelte Verwendung von (9.16) läßt sich formal als algebraisches Mehrgitterverfahren [2] interpretieren. Da sich der behandelte Ansatz als LU–Zerlegung für ein geeignet erweitertes System interpretieren läßt, können Ergebnisse für unvollständige Dreieckszerlegungen [7] auf den Fall der geschachtelten Divide & Conquer Strategie übertragen werden.

7. Zur Anwendung der geschachtelten Divide & Conquer Strategie lassen sich Zerlegungen der Ausgangsmatrix in einen blockdiagonalen Anteil und eine Restmatrix von niedrigem Rang gut einsetzen. Hier konnte gezeigt werden, daß sich durch geeignete Modifikationen des blockdiagonalen Anteils Strukturen der Ausgangsmatrix, wie etwa Symmetrie und M–Matrixeigenschaft auf den blockdiagonalen Anteil wie auch das Kopplungssystem übertragen. Weiter läßt sich die Restmatrix der Zerlegung auf eine sehr einfache Weise faktorisieren. Dadurch sind auf einfache Weise die Voraussetzungen für den Einsatz von (9.16) ermöglicht.

8. Beim Einsatz blockdiagonaler Zerlegungen lassen sich Modifikationen des blockdiagonalen Anteils durchführen, welche darauf abzielen, die Eigenschaften des Kopplungssystems zu verbessern. Diese Modifikationen, eine Art algebraischer Randbedingung für die Diagonalblöcke, können auf approximative blockdiagonale Lösungen von Riccatigleichungen zurückgeführt werden. Während hier im allgemeinen noch gekärt werden muß, welche blockdiagonalen Lösungen am besten geeignet sind, die Lösung der Riccatigleichung zu approximieren, konnte im symmetrischen Fall sogar die Optimalität im Sinne quadratischer Formen nachgewiesen werden.

9. Die Verwendung blockdiagonaler Zerlegungen führt auf eine natürliche parallele Verteilung der Matrix und insbesondere des Kopplungssystemes S_c aus (9.16), welches eine Übertragung des Konzeptes addierender und überlappender Vektoren [8],[7] aus den Gebietszerlegungsmethoden partieller Differentialgleichungen auf den algebraischen Fall ermöglicht. Somit steht ein leichter Zugang zur parallelen Behandlung von S_c zur Verfügung.

10. Die parallele Behandlung des Kopplungssystems für den Fall der geschachtelten Divide & Conquer Strategie läßt sich durch Bündeln der einzelnen Aufdatierungen zu einem einzigen Block verallgemeinern. Hierdurch läßt sich einerseits der parallele Einsatz der geschachtelten Divide & Conquer Strategie auf die ursprüngliche Ausgangsmatrix und ihr Kopplungssystem zurückführen. Andererseits bewirkt die Bündelung der hierbei auftretenden Modifikationen vom niedrigen Rang eine Reduktion der Kommunikationsschritte.

Bibliography

[1] S. Bondeli: Parallele Algorithmen zur Lösung tridiagonaler Gleichungssysteme. Dissertationsschrift, ETH Zürich, 1991.

[2] W. Dahmen, L. Elsner: Hierarchical Iteration. Lecture Notes on Numerical Fluid Dynamics, Hackbusch (ed.), Vieweg, 1988.

[3] Michael Eiermann: Fields of Values and Iterative Methods. Lin. Alg. and its Appl. **180** (1993), pp.167–197.

[4] R.W. Freund, G.H. Golub, N.M. Nachtigal: Iterative Solution of Linear Systems. Acta Numerica (1992), pp. 1–44.

[5] K.A. Gallivan, M. T. Heath, E. Ng, J.M. Ortega, B.W. Peyton, R.J. Plemmons, C. H. Romine, A.S. Sameh, R.G. Voigt: Parallel Algorithms for Matrix Computations. SIAM Philadelphia, 1990.

[6] G.H. Golub, C.F. Van Loan: Matrix Computations, second. ed. The Johns Hopkins University Press, 1989.

[7] G. Haase, U. Langer, and A. Meyer: The Approximate Dirichlet Domain Decomposition Method. Part I: An Algebraic Approach. Computing **47** (1991), pp.137–151.

[8] K.H. Law: A parallel finite element solution method. Computer and Structures, **23(6)**, (1989), pp.845–858.

[9] V. Mehrmann: Divide & Conquer Methods for Block Tridiagonal Systems. Parallel Computing **19** (1993), pp. 257–279.

[10] Y. Saad: Krylov subspace methods on supercomputers. SIAM J. Sci. Stat. Comput. **10** (1989), pp.1200-1232.

[11] Y. Saad, and M. Schultz: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Computing **7**(1986), pp.856–869.

[12] L. N. Trefethen: Pseudospectra of matrices. In: D. F. Griffiths and G. A. Watson, Numerical Analysis 1991, Longman Sci. Tech. Publ., 1992, pp.234–266.

Lebenslauf

Matthias Bollhöfer
Kanzlerstr. 42
09112 Chemnitz

Geboren am 14.05.1966 in Herford.

Staatsangehörigkeit deutsch.

Familienstand verheiratet, ein Kind.

Von August 1972 bis Juli 1976 Besuch der Grundschule am Elkenbreder Weg in Bad Salzuflen.

Von September 1976 bis Mai 1985 Besuch des Gymnasiums Hermannstraße bzw. nach dessen Umzug Gymnasium im Schulzentrum Aspe in Bad Salzuflen. Abschluß Abitur.

Von Oktober 1985 bis März 1992 Studium Mathematik mit Nebenfach Informatik an der Fakultät für Mathematik der Universität Bielefeld. Abschluß Diplom.

Von April 1992 bis März 1993 Promotionsstudium an der Fakultät für Mathematik der Universität Bielefeld.

Von April 1989 bis September 1991 sowie Oktober 1992 bis März 1993 regelmäßige Tutorentätigkeit an der Fakultät für Mathematik der Universität Bielefeld. Von April 1992 bis Juli 1992 Lehrbeauftragter an der Fachhochschule Bielefeld.

Seit April 1993 wissenschaftlicher Mitarbeiter am Fachbereich bzw. an der Fakultät für Mathematik der Technischen Universität Chemnitz. Von 1993 bis 1995 Mitarbeit in der Forschergruppe 'Scientific Parallel Computing' an der TU Chemnitz–Zwickau. Seit 1996 Mitarbeit im Teilprojekt 'Algebraische Zerlegungsmethoden' im Rahmen des Sonderforschungsbereiches 393 'Numerische Simulation auf massiv parallelen Rechnern' an der TU Chemnitz.