Universität Karlsruhe
Rechenzentrum
76128 Karlsruhe
Germany

Interner Bericht Nr. 74/99

# Properties of Approximate Inverses and Adaptive Control Concepts for Preconditioning

Author: Claus Koschinski



Datum: 16. Dezember 1999

# Properties of Approximate Inverses and Adaptive Control Concepts for Preconditioning

Zur Erlangung des akademischen Grades eines

**Doktors der Naturwissenschaften**

von der Fakultät für Mathematik der
Universität Karlsruhe

genehmigte

**Dissertation**

von

Dipl.-Math. Claus Peter Koschinski
aus Essen

Tag der mündlichen Prüfung: 15. Dezember 1999

Referent: Priv. Doz. Dr. R. Weiss

Korreferent: Prof. Dr. W. Schönauer

**Abstract**

In this work, we consider projection methods – a class of methods for calculating approximate inverses of general Hermitian and non-Hermitian matrices – and we consider the properties of these approximate inverses applied as explicit preconditioners of Krylov subspace methods.
We develop a theoretical framework for general projection methods. This framework includes an explicit representation of approximate inverses given by projection methods for given sparsity patterns, and, importantly, a statement on the quality of the approximation of such an approximate inverse to the exact inverse in form of a minimization property.
We introduce two strategies for the adaptive generation of sparsity patterns for general projection methods. Since these strategies depend on parameters, they are tunable with regards to the available computer architecture and to the characteristics of the considered linear system. Further, with the adaptive pattern derivation, projection methods can be considered as iterative methods for approximating the exact inverse. Thus, if necessary, a once calculated approximate inverse can be improved easily.
Based on the general definition of projection methods, we derive three new explicit preconditioning techniques – namely `L^TL-projection`, `LU-projection` and `Plain projection` – that include the adaptive generation of the sparsity pattern for the corresponding approximate inverses. Importantly, these new preconditioning techniques are inherently parallel, and hence well suited for today's supercomputers. We compare the performance of our new preconditioning techniques to both implicit and explicit state-of-the-art preconditioning methods. The results of these numerical experiments indicate that our new preconditioning techniques are competitive to the state-of-the-art preconditioning techniques as regards convergence rates. Since the three new preconditioning techniques can be parallelized easily, they will perform better on parallel computers. However, this investigation is beyond the scope of this thesis.

# Danksagung

Diese Dissertation entstand zum grössten Teil während meiner Tätigkeit am Rechenzentrum der Universität (TH) Karlsruhe.

Meine große Dankbarkeit gilt Priv. Doz. Dr. Rüdiger Weiss, der das Thema anregte, und der das Fortschreiten der Dissertation mit großem Interesse verfolgte. Die zahllosen mit Rüdiger Weiss geführten Diskussionen und seine vielen Anregungen waren mir während der gesamten Zeit der Anfertigung der Dissertation eine große Hilfe.

Prof. Dr. W. Schönauer danke ich für die freundliche Übernahme des Korreferats, sowie für zahlreiche Vorschläge zum Inhalt und zur Darstellung der Dissertation.

Ich danke Dipl.-Inf. W. Fries, der mir während meiner Tätigkeit in der Netzwerke-Abteilung des Rechenzentrums der Universität (TH) Karlsruhe ein geduldiger und ermutigender Chef war.

Mein Dank gilt Dr. Thomas Zeggel, der mit seinem scharfen Blick und großer Geduld zur Klarheit der formalen und inhaltlichen Struktur beigetragen hat. Simone Ritter danke ich für ihre Hilfe zur englischen Sprache, insbesondere was die Interpunktion angeht. Ich danke Dipl.-Ing. Jörg Hagenlocher für seine Hilfsbereitschaft und Gastfreundschaft.

# Contents

# 1 Introduction

Many problems in engineering, physics and chemistry can be formulated by systems of partial differential equations. The discretization of those equations with the finite difference or the finite element method brings forth large sparse linear systems of the form $Ax = b$, with $A \in I\!R^{n \times n}$ and $x,\ b \in I\!R^n$. The solution of such sparse linear systems is one of the key problems in today's scientific computing research.

Direct factorization methods, like e.g. the Gaussian elimination, are not recommendable for the solution of large sparse linear systems, since during such a computation the sparse structure of the coefficient matrix is destroyed and an inadmissible amount of computer memory is required.

In the last decades, a lot of research has been devoted to the construction of iterative solvers for linear systems and many methods have been proposed. Most state-of-the-art iterative solvers for linear systems belong to the family of Krylov subspace methods. One favorable property of Krylov subspace methods is that – apart from the coefficient matrix $A$ – only a very small amount of computer memory is additionally needed. The computational complexity of Krylov subspace methods is dominated by matrix-vector multiplications – operations that can be implemented very efficiently on today's vector and parallel computers.

A severe drawback of iterative methods for solving linear systems is their lack of robustness: the iterative solution process may fail to converge or be very slow. For Krylov subspace methods, the convergence speed of the iterative solution depends – by a rule of thumb – on the spectral properties of the coefficient matrix $A$, which is unknown in practical applications.

In the last years numerous strategies to enhance the robustness of Krylov subspace methods have been proposed. The concept of those preconditioning strategies is to consider a more easily solvable, related linear system instead of the original linear system $Ax = b$. Although all of these preconditioning strategies were found to be more or less successful for some problems, no stable, all-purpose preconditioning strategy is known up to now.

In general, all preconditioning methods can be classified as being either implicit or explicit. Implicit preconditioning methods are characterized by requiring additional equation solves in each step of the applied Krylov subspace method. Those preconditioning methods are successful for a relatively wide range of problems and computable in a small set-up time. On the other hand, implicit preconditioning methods are inherently sequential, and thus not well suited for today's vector and parallel computers. Explicit methods are characterized by requiring only one or more additional matrix-vector multiplications in each step of the applied Krylov subspace method. Those preconditioning methods seem to be less stable than the implicit methods. On the other hand, they offer a strategy dependent degree of parallelism in the set-up phase and full parallelism during the iterative solve.

In this work, we consider projection methods, which form a general class of methods for calculating approximate inverses and we consider the theoretical and the practical properties of preconditioning iterative methods with such approximate inverses. We develop a new theoretical framework and we present new adaptive strategies for deriving sparsity patterns for approximate inverses given by projection methods. Furthermore, we propose three new preconditioning techniques, which essentially are particular projection methods with adaptive pattern derivation, and we consider the practical and numerical properties of these new algorithms by comparing them to the state-of-the-art preconditioning techniques.
This work is organized in the following way:
In chapter 2, we summarize the notation used throughout this work.
We begin chapter 3 by describing the general concept of iterative solvers for linear systems. In section 3.2, we discuss the general concept of enhancing the efficiency and the robustness of iterative solvers by preconditioning. Many of the state-of-the-art iterative solvers for linear systems belong to the family of orthogonalization methods. This family of iterative methods is described in section 3.3 and statements on the convergence of such iterative methods (with and without preconditioning) are given. In section 3.4 Residual-Minimizing smoothing, a helpful tool for observing the convergence of iterative methods, is described. The family of Krylov subspace methods – a subclass of orthogonalization methods – is described in section 3.5. The iterative solvers considered in our numerical tests belong to the family of Krylov subspace methods. We describe these iterative solvers in greater detail and we give a pseudo-code formulation of these solvers containing preconditioning.
We consider the theoretical properties of projection methods in chapter 4. At the beginning of section 4.1, we give a bundle of definitions and nomenclature for the treatment of sparse vectors and matrices, and the corresponding sparsity patterns. These technical terms for sparse vectors and matrices are employed in the remainder of this work. Then we state the definition of projection methods and we discuss the basic concept behind projection methods and their application for preconditioning iterative solvers. We deduce criteria for the existence and formulae for the explicit representation of approximate inverses given by projection methods. Further, we state the explicit calculation of approximate inverses by projection methods in form of pseudo-code algorithms. Importantly, these pseudo-code algorithms are inherently parallel, since the major part of their computational complexity consists in solving many small independent linear systems.
In section 4.2, we develop statements on the quality of the approximation of an approximate inverse calculated by a projection method to the corresponding exact inverse. These statements compare the columns of the approximate inverse to the corresponding columns of the exact inverse; the difference between two such columns is measured in form of a columnwise minimization property in a subspace with a dimension equal to the number of non-zeros allowed in the corresponding column of

the approximate inverse. For illustrating the potential of the theoretical framework for projection methods developed in sections 4.1 and 4.2, we conclude section 4.2 with considering two particular projection methods and their theoretical properties in greater detail.

The design of strategies for the adaptive derivation of sparsity patterns for approximate inverses calculated by projection methods is considered in chapter 5. In practical applications, the shape of the sparsity pattern utilized for calculating an approximate inverse given by a projection method is decisive for the resulting acceleration of the preconditioned iteration. Since in general practice no information on a suitable sparsity pattern for an approximate inverse given by a projection method is available, the issue of adaptively determining a suitable sparsity pattern is of great importance.

In section 5.1, we give an outline on the general design of the new adaptive pattern derivation strategies for projection methods, and we discuss strategies for controlling the way of selecting the adaptive sparsity pattern as well as controlling the maximum amount of non-zeros in the resulting approximate inverse. The basic idea behind these pattern derivation strategies rests on augmenting a given sparsity pattern of the approximate inverse. The indices added to the given sparsity pattern are selected in such a way that the approximation statements given in section 4.2 yield the best possible approximation of the augmented approximate inverse to the exact inverse. For this purpose, auxiliary quantities – namely the decrease rates – are calculated. For reasons of simplicity, we postpone the actual calculation of these decrease rates to sections 5.2 and 5.3. Based on the general considerations in section 5.1, we present and discuss two new algorithms for the adaptive pattern derivation for projection methods in a pseudo-code formulation, and we consider ways for choosing the parameters, which control these algorithms, with regards to the available computer hardware and to problem dependent specifics. Importantly, these new adaptive pattern derivation algorithms are inherently parallel, and hence appealing for today's supercomputers.

In sections 5.2 and 5.3, we consider two approaches for the calculation of the decrease rates, which are required for the adaptive pattern derivation algorithms presented in section 5.1.

In chapter 6, we consider practical preconditioning techniques. We describe some of the state-of-the-art preconditioning methods, and we introduce three new preconditioning techniques, which are projection methods with adaptive pattern derivation.

In section 6.1, we discuss the concept of normalization. Two variants of normalization are classified in terms of projection methods.

In section 6.2, the Frobenius norm minimizing preconditioning approach is considered. This preconditioning method has been classified in terms of projection methods by Zimmermann in [83]. This preconditioning technique is, since it is a projection method, inherently parallel, and much attention has been devoted to

refining this method in the last years (see e.g. [17], [20], [32] and [39]).
In section 6.3, we propose a new preconditioning technique – the `Plain projection` algorithm. This preconditioning technique is based on a method proposed by Kolotilina and Yeremin in [46], which we classify in terms of projection methods, combined with the adaptive sparsity pattern derivation developed in chapter 5.1. Importantly, the `Plain projection` algorithm is fully parallelizable.
The concepts of the families of incomplete $LU$-decompositions and incomplete Cholesky-decompositions are briefly described in sections 6.4 and 6.5. These families of methods have been refined for decades, and numerous variants have been proposed. Both approaches are known to be quite robust and efficient, but unfortunately the resulting algorithms are inherently sequential, and are thus not well-suited for parallel environments.
In section 6.6, we describe a preconditioning technique recently proposed by Benzi, Meier and Tůma for symmetric positive definite matrices, and by Benzi and Tůma for non-symmetric matrices. This preconditioning technique is based on an incomplete $A$-Biconjugation process. The resulting algorithm is called `AINV`. On output, the `AINV` algorithm furnishes approximate inverses for the triangular factors of the Cholesky-decomposition, or the $LU$-decomposition of the coefficient matrix, respectively. Recent results of a parallel implementation of the `AINV` algorithm published in [5] indicate that this preconditioning approach, although sequential in principle, can efficiently be implemented in a parallel environment.
In section 6.7, we propose two new preconditioning techniques, the $\mathtt{L^TL}$`-projection` algorithm and the `LU-projection` algorithm. With the $\mathtt{L^TL}$`-projection` algorithm the inverse of a triangular factor of the Cholesky-decomposition of a hermitian positive definite coefficient matrix is approximated. The `LU-projection` algorithm approximates the inverses of the triangular factors of the $LU$-decomposition of the coefficient matrix. Interestingly, for both algorithms no information on the actual shape of the triangular factors is required. Essentially, both algorithms rest on the `FSAI` preconditioning technique proposed by Kolotilina and Yeremin in [46] for fixed sparsity patterns. We classify the methods of Kolotilina and Yeremin in terms of projection methods, and combine them with the adaptive pattern derivation strategies developed in chapter 5.1. The resulting $\mathtt{L^TL}$`-projection` algorithm and the `LU-projection` algorithm are inherently parallel.
In chapter 7, we compare the state-of-the-art preconditioning methods with our new approaches – the `Plain projection` algorithm, the $\mathtt{L^TL}$`-projection` algorithm and the `LU-projection` algorithm – by applying them to numerous symmetric and non-symmetric test problems.
In chapter 8, we comprise the results of our numerical tests, and we give an outlook on open questions.

# 2 Notation

In this section, we introduce the notation that is used throughout this paper.

**Letters**
We denote integers and vectors by lower-case Latin letters and matrices by capital Latin letters. The letter $n$ always denotes an element of $\mathbb{N}$. For notational convenience we define the field $\mathbb{K}$ by $\mathbb{K} \in \{\mathbb{R},\mathbb{C}\}$.
The entries of a matrix $A \in \mathbb{K}^{n\times n}$ are designated by $a_{ij}$. Real or complex scalars are denoted by lower case Greek letters. The real part of a complex number $\gamma$ is denoted by $\mathrm{Re}\,(\gamma)$.

**Matrices**
The transposed of a matrix $A \in \mathbb{K}^{n\times n}$ is denoted by $A^T$ and the conjugated transposed is denoted by $A^H$. By $\rho(A)$ we denote the spectral radius of a $(n \times n)$-matrix $A$, which is defined as

$$\rho(A) := \max_{1\le i\le n} \mid \lambda_i \mid,$$

where $\lambda_i$ for $i = 1,\ldots,n$ are the eigenvalues of $A$.
For a tridiagonal matrix $A \in \mathbb{K}^{n\times n}$ with the main diagonal $a_{i,i}$ for $i = 1,\ldots,n$ and with the left-hand and right-hand side neighbouring diagonals $a_{i+1,i}$ and $a_{i,i+1}$ for $i = 1,\ldots,n-1$, we write in abbreviation

$$A = tridiag\,(a_{i+1,i}, a_{i,i}, a_{i,i+1})\,.$$

We call a matrix $A \in \mathbb{K}^{n\times n}$ structural symmetric, if the relation

$$a_{i,j} \neq 0 \Longleftrightarrow \quad a_{j,i} \neq 0$$

holds for $i, j = 1,\ldots,n$.
The $(n \times n)$-identity matrix is denoted by $I$, the $i$-th unit vector in $\mathbb{K}^n$ by $u_i$.
The symmetric part of an arbitrary real square matrix is defined by

$$M_A := \frac{1}{2}(A + A^T),$$

and its skew-symmetric part by

$$N_A := \frac{1}{2}(A - A^T).$$

For notational convenience, we introduce some not strictly mathematical nomenclature:
We call a matrix $A \in \mathbb{K}^{n\times n}$ dense , if only a few elements of $A$ are zero. Conversely, we call a matrix sparse, if only relatively few elements of $A$ differ from zero.

Let the matrices $B_L, B_R \in \mathbb{K}^{n\times n}$ be such that the product matrix $B_R B_L$ is an approximation to the inverse $A^{-1}$ of a non-singular matrix $A \in \mathbb{K}^{n\times n}$, i.e.

$$\begin{aligned} & B_L A B_R && \approx I \\ \iff \quad & B_R B_L && \approx A^{-1}. \end{aligned}$$

Then, we call $B_R B_L$ approximate inverse of $A$. If $B_L = I$ (or $B_R = I$), we call $B_R$ right-hand (or $B_L$ left-hand) side approximate inverse of $A$.

**Definiteness**
A Hermitian matrix $A \in \mathbb{K}^{n\times n}$ is called positive (negative) definite, if $x^T A x > 0$ ($x^T A x < 0$) holds for all $x \neq 0,\ x \in \mathbb{K}^n$.
We refer to a non-symmetric matrix $A \in \mathbb{R}^{n\times n}$ as positive (negative) real, if its symmetric part is positive (negative) definite.

**Vector Norms / Dot-Products**
Let the matrix $A \in \mathbb{K}^{n\times n}$ be Hermitian positive definite, then for the vector $x \in \mathbb{K}^n$ the quantity

$$\|x\|_A := (x^H A x)^{\frac{1}{2}}$$

is a norm on $\mathbb{K}^n$. This norm is called $A$-norm of the vector $x$. In particular, we have $\|x\|_I = \|x\|_2$, where $\|.\|_2$ denotes the Euclidian norm.
If $A \in \mathbb{R}^{n\times n}$ is positive real and $M_A$ is the symmetric part of $A$, we write $\|x\|_A$ in abbreviation of $\|x\|_{M_A}$.
For an indefinite matrix $A \in \mathbb{K}^{n\times n}$ and the vector $x \in \mathbb{K}^n$, the quantity $\|x\|_A^2$ is the mnemonic abbreviation of $x^H A x$.

**Matrix Norms**
Let the matrix $B \in \mathbb{K}^{n\times n}$ be Hermitian positive definite. Then the matrix norm induced by the vector norm $\|.\|_B$ is defined by

$$\|A\|_B := \sup_{x\neq 0} \frac{\|Ax\|_B}{\|x\|_B},$$

where $A$ is a matrix in $\mathbb{K}^{n\times n}$ and the vector $x$ is in $\mathbb{K}^n$.
In particular for $A \in \mathbb{K}^{n\times n}$ for the matrix norm induced by the Euclidian norm, the identity

$$\|A\|_2 = \sqrt{\max_{1\leq i\leq n} \lambda_i}$$

holds, where the $\lambda_i$ denote the eigenvalues of $A^H A$ for $i = 1, \ldots, n$.

The Frobenius norm of a matrix $A \in \mathbb{K}^{n\times n}$ is defined by

$$\left\|A\right\|_F := \left(\sum_{i=1}^{n} \left\|Au_i\right\|_2^2\right)^{\frac{1}{2}}.$$

We denote the condition number of a matrix $A \in \mathbb{K}^{n\times n}$ by

$$\kappa(A) := \left\|A\right\|_2 \left\|A^{-1}\right\|_2,$$

where $\left\|A\right\|_2$ denotes the Euclidian matrix norm.

# 3 Linear Solvers and Preconditioning

In this chapter, we give a brief sketch of iterative methods for solving linear systems, and we describe the general concept for preconditioning those iterative solvers.
In section 3.1, we give the general approach and some nomenclature for iteratively solving linear systems.
The general concept for preconditioning iterative solvers is sketched in section 3.2. Furthermore, some demands for the design of efficient preconditioning methods are considered. For ample surveys on iteratively solving linear systems and preconditioning see e.g. [11], [12], [54], [62] and [80].
In section 3.3, we describe orthogonalization methods (see [80]), a general class of iterative methods for solving linear systems, which provides us with a theoretical framework for handling iterative methods.
A helpful tool for controlling the convergence of iterative solvers – the Residual-Minimizing smoothing algorithm – is described in section 3.4.
In section 3.5, we give a brief introduction to Krylov subspace methods, and we shortly describe four particular Krylov subspace methods for solving both symmetric and non-symmetric linear systems. We performed our numerical tests of preconditioning strategies (presented in chapter 7) using those four iterative solvers.

## 3.1 Iterative Solvers for Linear Systems

In this section, we give a brief description of the iterative solution of the linear system

$$Ax = b, \tag{3.1}$$

where $A$ is a non-singular matrix in $\mathbb{K}^{n\times n}$, and $x$ and $b$ are vectors in $\mathbb{K}^n$. The vector $x$ is the desired solution of the linear system with the coefficient matrix $A$ and the right-hand side vector $b$.
Starting with an initial guess $x_0 \in \mathbb{K}^n$, iterative solvers generate a sequence $x_k \in \mathbb{K}^n$, with $k \in \mathbb{N}$, of approximation vectors. The construction of a new approximation involves one or more preceeding approximations, as well as the matrix $A$ and the right-hand side vector $b$. Formally, the construction of the $k$-th approximation can be written as

$$x_k := f_k(x_0, \dots, x_{k-1}, A, b)$$

with a function $f_k : \mathbb{K}^{n\times(n+k+1)} \to \mathbb{K}^n$. In theory, the functions $f_k$ are chosen such that the sequence $(x_k)_{k\in\mathbb{N}}$ converges towards the exact solution $x$, i.e.

$$x_k \to x \text{ for } k \to \infty.$$

In practice, the iteration process is stopped after a finite number of steps when a sufficient approximation is reached. The natural choice for the stopping criterion would be based on the norm of error $e_k \in I\!K^n$, defined by

$$e_k := x_k - x, \tag{3.2}$$

where $x$ denotes the exact solution of the linear system (3.1). If the norm of the error becomes smaller than a prescribed threshold, the iteration is stopped.
Since in practice the errors $e_k$ are not known, the norm of the residuals $r_k \in I\!K^n$, defined by

$$r_k := Ax_k - b, \tag{3.3}$$

are used as stopping criterion, i.e. the iteration is stopped when the norm of the residuals becomes smaller than a threshold value.
Errors and residuals are connected by

$$r_k = Ae_k. \tag{3.4}$$

The following considerations show the connection of the Euclidian norms of the residuals and errors. By equation (3.4) we have

$$\|r_k\|_2 = \|Ae_k\|_2 \leq \|A\|_2 \cdot \|e_k\|_2$$

and

$$\|e_k\|_2 = \|A^{-1}r_k\|_2 \leq \|A^{-1}\|_2 \cdot \|r_k\|_2 .$$

With the two statements above, the following inequalities hold:

$$\frac{1}{\kappa}\frac{\|r_k\|_2}{\|r_0\|_2} \leq \frac{\|e_k\|_2}{\|e_0\|_2} \leq \kappa\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa^2\frac{\|e_k\|_2}{\|e_0\|_2}, \tag{3.5}$$

where $\kappa$ denotes the condition number of the matrix $A$. If the condition number of $A$ is small, the norms of the residuals and errors are strongly connected by inequalities (3.5). In this case, the reduction of the norm of the residuals implies that the norm of the errors decreases as well. If the condition number of $A$ is large, the connection of the norms of the residuals and errors is weak, and it is possible that, although the norms of the residuals decrease, the norms of the errors may stagnate, or even grow. In this case, the iterative method may fail to produce an acceptable approximation to the solution of the linear system.

By the considerations above, the general concept of iterative methods for linear systems is briefly expounded. In the following section, we describe the basic idea of enhancing both the speed of convergence and the robustness of iterative solvers for linear systems by preconditioning.
In sections 3.3 – 3.5, we consider general classes of iterative solvers and we give theoretical and practical considerations on preconditioning those iterative methods.

## 3.2 Preconditioning Iterative Solvers

A severe drawback of iterative solvers for linear systems is their lack of robustness in contrast to direct solvers. If the iterative solver fails to converge, no approximate solution for the linear system (3.1) is obtained. This situation is unacceptable in practice.

The objective of preconditioning iterative solvers is to remedy this problem by enhancing both their efficiency and their robustness.

**Basic Concept**

In general, for preconditioning the original linear system $Ax = b$ from (3.1) two non-singular matrices $P_L,\ P_R \in I\!\!K^{n\times n}$ are determined such that the preconditioned linear system

$$P_L A P_R y = P_L b, \text{ with } x = P_R y, \tag{3.6}$$

is easier solvable by an iterative method than the original linear system, i.e. the general strategy of preconditioning iterative solvers is:

1. Choose the preconditioning matrices $P_L$ and $P_R$.
2. Solve the preconditioned system $P_L A P_R y = P_L b$ iteratively. This yields the approximate solution $y_k$ of the preconditioned linear system (3.6).
3. Obtain the desired approximate solution $x_k$ of the original system (3.1) by $x_k := P_R y_k$.

The matrix $P_L$ is called left-hand and $P_R$ is called right-hand side preconditioning matrix. If $P_L \neq I$ and $P_R \neq I$, we talk of two-sided preconditioning; if $P_L$ equals the identity matrix, we talk of right side, and if $P_R$ equals the identity matrix, we talk of left side preconditioning. In practice, preconditioning from the left, from the right and from both sides is applied.

**Theoretical Properties**

The residuals $\bar{r}_k \in I\!\!K^n$ and the errors $\bar{e}_k \in I\!\!K^n$ of an iterative method applied to the preconditioned linear system (3.6) are by (3.2) and (3.3):

$$\bar{r}_k := P_L A P_R y_k - P_L b \tag{3.7}$$

and

$$\bar{e}_k := y - y_k, \tag{3.8}$$

where $y_k$ denotes the iterates of the iterative method applied to the preconditioned linear system (3.6).

The following definition establishes the connection between the iterative method applied to the preconditioned linear system (3.6) and the desired approximate solution of the original linear system (3.1).

**Definition 3.1** (Induced Iteration)
*Let $y_k \in I\!K^n$ denote the iterates of an iterative method applied to the preconditioned linear system* (3.6)*. Then, by*

$$x_k := P_R y_k \tag{3.9}$$

*for $k \geq 1$, a sequence of approximate solutions of the original linear system $Ax = b$ from* (3.1) *is defined. In this sense, applying an iterative method to the preconditioned linear system* (3.6) **induces an iterative method** *for the original linear system* (3.1)*.*

The following lemma connects the residuals and errors of the preconditioned iteration and the induced iteration for the original linear system.

**Lemma 3.2**
*Let the residuals and errors of an iterative method applied to the preconditioned linear system* (3.6) *be denoted by $\bar{r}_k$ and $\bar{e}_k$. Then, the following equations hold:*

$$\bar{r}_k = P_L r_k, \tag{3.10}$$

*and*

$$\bar{e}_k = P_R^{-1} e_k, \tag{3.11}$$

*where $r_k$ and $e_k$ denote the residuals and errors of the induced iterative method for the original linear system* (3.1)*.*

**Proof.**

We give the proof for the assertions, although they are well known (see e.g. [80]), to enlighten the relation between the residuals and errors of the original and the preconditioned linear system.
Let $x$ and $y$ denote the solution of the original linear system (3.1) and the preconditioned linear system (3.6), respectively. By (3.7) and (3.8) we have

$$\begin{aligned} \bar{r}_k &\overset{(3.7)}{=} P_L A P_R y_k - P_L b \\ &\overset{(3.9)}{=} P_L A x_k - P_L b \\ &= P_L r_k, \end{aligned}$$

and

$$
\begin{aligned}
\bar{e}_k &\overset{(3.8)}{=} y - y_k \\
&\overset{(3.9)}{=} P_R^{-1}x - P_R^{-1}x_k \\
&= P_R^{-1}e_k.
\end{aligned}
$$

$\diamondsuit$

The crucial point to preconditioning is the choice of the preconditioning matrices $P_L$ and $P_R$. The optimal choice would be

$$P_R P_L := A^{-1},$$

since then, with $A = P_L^{-1}P_R^{-1}$, the desired solution $x$ of the original linear system (3.1) would be known at once by

$$y = P_L A P_R y = P_L \left(P_L^{-1}P_R^{-1}\right) P_R y \overset{(3.6)}{=} P_L b$$

and

$$x = P_R y = P_R P_L b = A^{-1}b.$$

The choice $P_R P_L := A^{-1}$ is not of practical relevance, since explicitely calculating $A^{-1}$ is a far more difficult problem than solving the original linear system (3.1).
However, most preconditioning strategies intend to determine $P_L$ and $P_R$ such that $P_L A P_R$ is closer to the unit matrix than the original matrix $A$ in some sense. Thus, for those methods the matrix $P_R P_L$ is an approximate inverse of $A$.
The residuals $\bar{r}_k$ and errors $\bar{e}_k$ of an iterative method applied to the preconditioned linear system (3.6) are connected by

$$\bar{r}_k = P_L A P_R \bar{e}_k.$$

Analogously to (3.5), the Euclidian norms of the residuals and errors of the preconditioned system are connected by the following inequalities:

$$\frac{1}{\kappa_p}\frac{\left\|\bar{r}_k\right\|_2}{\left\|\bar{r}_0\right\|_2} \leq \frac{\left\|\bar{e}_k\right\|_2}{\left\|\bar{e}_0\right\|_2} \leq \kappa_p \frac{\left\|\bar{r}_k\right\|_2}{\left\|\bar{r}_0\right\|_2} \leq \kappa_p^2 \frac{\left\|\bar{e}_k\right\|_2}{\left\|\bar{e}_0\right\|_2}, \tag{3.12}$$

where $\kappa_p$ denotes the condition number of the matrix $P_L A P_R$. Thus, if the inverse of $A$ is approximated by $P_L$ and $P_R$, so that the condition number of $P_L A P_R$ is smaller than the condition number of the original coefficient matrix $A$, the residuals and errors of the preconditioned system are connected more strongly than the ones of the original system (3.1).

Some preconditioning methods, for instance the `SPAI`-methods (see section 6.2), furnish only one-sided, i. e. either left-hand side or right-hand side, approximate inverses. But in general applications, it is not known whether a left-hand side or a right-hand side approximate inverse is favorable to the other.
The following theorem from [53] states on the difference between calculating right-hand side and left-hand side approximate inverses of non-singular matrices in $\mathbb{R}^{n\times n}$.

**Theorem 3.3** ([53], p. 287)
*Let $\epsilon$ and $\kappa$ be two arbitrary positive real numbers. Then, for each $n \geq 2$ there exist non-singular $(n \times n)$-matrices $A$ and $X$ such that $XA$ has each of its elements differing from the corresponding element of the identity matrix by less than $\epsilon$ and such that $AX$ has all of its elements greater in absolute value than $\kappa$.*

The above theorem states, that a one-sided approximate inverse may be a good preconditioner for the one side while being a poor preconditioner for the other side. Further, in [53] it is shown, that if theorem 3.3 applies for a matrix $A \in \mathbb{R}^{n\times n}$ and a corresponding approximate inverse $X$ with small $\epsilon$ and large $\kappa$, then the matrix $A$ is ill-conditioned. Thus, for one-sided preconditioning, the side of preconditioning may be important. Therefore, at least in tendency, preconditioning methods that calculate two-sided approximate inverses may be more robust than methods which calculate only one-sided approximate inverses.

**Practical Considerations**
If the coefficient matrix $A$ of the original linear system (3.1) is Hermitian, the coefficient matrix $P_L A P_R$ of the preconditioned linear system (3.6) should be Hermitian as well, since in this case special iteration algorithms which exploit this property can be applied. This can easily be obtained by two-sided preconditioning with $P_L = P_R^H$.

Most of today's preconditioning approaches can be classified as being either implicit or explicit. A preconditioning approach is called implicit, if the preconditioning matrices $P_L$ and $P_R$ are not known explicitly, so that the application of the preconditioning matrix involves solving a supplemental linear system in each iteration step of the utilized iterative solver. Such preconditioning methods are for example incomplete decompositions like `ILU` (this method is briefly sketched in section 6.4, see e.g. [40], [82] for detailed surveys), `IC` (see section 6.5 for a brief description, see [48], [52], [76] for detailed surveys).
Conversely, a preconditioning approach is called explicit, if the preconditioning matrices $P_L$ and $P_R$ are known explicitly. Such methods are for instance the `SPAI`-method (see section 6.2 for a brief description, and e.g. [32], [39] for more details)

or the `AINV`-method (see section 6.6 for a brief description, and [8], [9] for detailed investigations). If the utilized iterative solver is a Krylov subspace method (see section 3.5), each iteration step of the chosen Krylov subspace method applied on the preconditioned linear system only requires additional matrix-vector multiplications, compared to the iteration steps of the chosen solver applied to the unpreconditioned linear system. Since matrix-vector multiplications are efficiently vectorizable and parallelizable, using explicit preconditioning with Krylov subspace methods is well suited for the implementation on today's supercomputers (this point is surveyed in greater detail in section 3.5).

In general, all preconditioning techniques are characterized by choosing particular strategies for answering the following two questions:

1. Which locations of the preconditioning matrices $P_L$ and $P_R$ should contain non-zero entries?

2. How should the values of the non-zero entries be computed?

Due to limited CPU-time and memory in practical applications, constraints to the maximum number of non-zeros in the preconditioning matrices $P_L$ and $P_R$ must be imposed. Thus, the matrix $P_R P_L$ in general is an approximate inverse of the matrix $A$, where $P_L$, $P_R$ and $A$ are from (3.6).
It is an open – and highly problem dependent – question whether or not the inverse $A^{-1}$ of a given non-singular matrix $A \in I\!\!K^{n\times n}$ can be approximated by an approximate inverse such that the convergence of the preconditioned iteration is substantially accelerated. If the inverse $A^{-1}$ has relatively few entries of "large" magnitude while all other entries are "small" or zero, it is likely – but by no means guaranteed – that an appropriate approximate inverse can be an efficient preconditioner. On the other hand, if the inverse $A^{-1}$ has "many" entries of the same size, it may be difficult to determine an approximate inverse of $A$, which is an effective preconditioner. The following lemma from [16] considers this problem for right-hand side approximate inverses:

**Lemma 3.4** ([16], pp. 15 –16)
*Let the matrix $A \in I\!\!K^{n\times n}$ be non-singular, and let the matrix $P \in I\!\!K^{n\times n}$ be an approximate inverse of $A$. Further, let $B := A^{-1}$, let the numbers $\tau_k \in I\!\!R$ be such that the relations*

$$\left\|(AP - I)\, u_k\right\|_1 \leq \tau_k \tag{3.13}$$

*hold for $k = 1, \ldots, n$, where $u_k \in I\!\!K^n$ denotes the $k$-th unit-vector, and let*

$$\tau := \max_{1\leq k\leq n} \tau_k. \tag{3.14}$$

*Then the following assertions hold:*

*i) If the inequality*

$$| b_{ij} | > \tau_j \max_{1 \le k \le n} | b_{ik} | \tag{3.15}$$

*holds for any element $b_{ij}$ of the matrix $B$, then the element $p_{ij}$ of the approximate inverse $P$ is non-zero.*

*ii) If the inequality*

$$| b_{ij} | > \tau \max_{1 \le k,t \le n} | b_{kt} | \tag{3.16}$$

*holds for all non-vanishing elements $b_{ij}$ of the matrix $B$, then the matrix-pattern of the sparse approximate inverse $P$ contains the matrix-pattern of the exact inverse $A^{-1}$. Thus, in this case, if $A^{-1}$ is dense, the approximate inverse $P$ is dense as well.*

The numbers $\tau_k$ defined in (3.13) form a column-wise measure of the quality of the approximation of the approximate inverse $P$ to the exact inverse $A^{-1}$ in the one-norm. The smaller the numbers $\tau_k$ are, the closer the approximate inverse $P$ to the exact inverse $A^{-1}$ is.

Assertion *i*) of lemma 3.4 states that the large entries of the exact inverse $A^{-1}$ in the sense of (3.15) indicate non-zero elements in the approximate inverse $P$.
Assertion *ii*) states that a good approximation of $P$ to the exact inverse $A^{-1}$, in the sense of a small number $\tau$ from (3.14), may require a dense approximate inverse $P$.

Because of CPU-time and memory restrictions in practice, all preconditioning approaches compromise between the conflicting goals of approximating $A^{-1}$ as good as possible and still preserving some kind of sparsity for the preconditioning matrices. Preserving the sparsity of $P_L$ and $P_R$ usually involves some dropping strategy or some restriction to the amount of fill-in. If a dropping strategy is applied, all entries of the preconditioning matrices with an absolute value smaller than some prescribed threshold are set to zero. If the amount of fill-in in the preconditioning matrices is restricted, the computation of the entries of the preconditioning matrices is stopped when the number of non-zeros in $P_L$ and $P_R$ has reached the prescribed limit.
As for the CPU-time of preconditioning techniques, to be of practical relevance the sum of the CPU-time for constructing the preconditioner and the CPU-time of the preconditioned iteration must be considerably less than the CPU-time for the unpreconditioned iteration.

In practical applications, it may be difficult to determine whether or not the preconditioning matrices $P_L$ and $P_R$, calculated with any preconditioning technique, accelerate the iterative method, when applied to the preconditioned linear system (3.6), sufficiently or are even non-singular. The following proposition from [16] considers this problem for right-hand side approximate inverses:

**Proposition 3.5** ([16], p. 14)
*Let the matrix $A \in \mathbb{K}^{n\times n}$ be non-singular, let the matrix $P \in \mathbb{K}^{n\times n}$ be an approximate inverse of $A$, and let the real number $\tau$ be defined by*

$$\tau := \max_{1\leq k\leq n} \left\| (AP - I)\, u_k \right\|_1 ,$$

*where $u_k$ denotes the $k$-th unit-vector in $\mathbb{K}^n$. Then the following assertions hold:*

*i) Any eigenvalue $\lambda \in \mathbb{C}$ of the matrix $AP$ is located in the disc*

$$\mid \lambda - 1 \mid < \tau. \tag{3.17}$$

*ii) If $\tau < 1$, then the matrix $AP$ is non-singular.*

*iii) If the columns $Pu_{j_1}, \ldots, Pu_{j_s}$ with $2 \leq s \leq n$, of the matrix $P$ are linearly dependent, then*

$$\left\| (AP - I)\, u_j \right\|_1 \geq 1 \tag{3.18}$$

*holds for at least one index $j \in \{j_1, \ldots, j_s\}$.*

Thus, if the number $\tau$ from (3.17) is less than one, the preconditioned linear system (3.6) has two favorable properties: the coefficient matrix is non-singular, and the eigenvalues of the coefficient matrix are clustered around one in the positive half-plane of $\mathbb{C}$. Unfortunately, in practical applications it may be impossible to determine the approximate inverse $P$ such that the number $\tau$ from inequality (3.17) is smaller than one, since this may force the approximate inverse $P$ to be too dense.

In this section, we have presented the basic concept of preconditioning iterative solvers, and we have stated theoretical and practical points on that matter. For supplemental discourses on iterative linear solvers and preconditioning, we refer to [11], [12], [54], [62] and [80].
In the remainder of this chapter, we consider iterative solvers in more detail. We present general classes of iterative methods, and we give remarks and the algorithmic formulation of four particular iterative methods for linear systems. Those are the iterative methods considered in our numerical tests on preconditioning (presented in chapter 7).

## 3.3 Orthogonalization Methods

In this section, we describe a general class of iterative methods for linear systems – the orthogonalization methods. Importantly, many of the state of the art iterative methods for linear systems, e.g. all Krylov subspace methods, are orthogonalization methods as well. Thus, the following theoretical results for orthogonalization methods furnish a unifying framework for many of the state of the art iterative solvers for linear systems.

The approximation theorem for orthogonalization methods given in this section provides some insight on the convergence behavior of iterative methods, and is hence the starting-point for the development of the preconditioning methods in the remainder of this work.
An ample overview and a classifying theory on orthogonalization methods is given in [80].
The following definition of orthogonalization methods was given in [80] for $\mathbb{K} = \mathbb{R}$. We extend it to the complex case:

**Definition 3.6** (Orthogonalization Method, [80], p. 29)
*Let the matrix $A$ in $\mathbb{K}^{n\times n}$ be non-singular and let $x$ and $b$ denote vectors in $\mathbb{K}^n$. An iterative method for the solution of the linear system $Ax = b$ with the iterates $x_k \in \mathbb{K}^n$ is called* **orthogonalization method**, *if the following relations hold:*

*For $k \geq 1$*

$$x_k \in \tilde{x}_k + \operatorname{span}\left(q_{k-l_k,k}, \ldots, q_{k-1,k}\right), \tag{3.19}$$

*where $l_k \leq k$, $\tilde{x}_k \in \operatorname{span}(x_{k-l_k}, \ldots, x_{k-1})$, $q_{k-i,k} \in \mathbb{K}^n$ for $i = 1, \ldots, l_k$, and for some auxiliary non-singular matrix $Z_k \in \mathbb{K}^{n\times n}$ the orthogonality condition*

$$r_k^H Z_k q_{k-i,k} = 0 \tag{3.20}$$

*is satisfied for $i = 1, \ldots, l_k$.*

*The vectors $q_{k-i,k}$ are called* **search directions**, *and the matrices $Z_k$ are called* **orthogonalization matrices**.
*Let $l_{res}$ and $l_{max}$ be positive integers. The orthogonalization method is called*

- **exact**, *if $l_k = k$, i. e. all $k$ search directions are used for the calculation of the new iterate,*

- **restarted**, *if $l_k = (k-1) \mod l_{res}$ with $l_{res} \in \mathbb{N}$, i. e. periodical restarts are made after $l_{res}$ steps,*

- **truncated**, *if* $l_k = \min(k, l_{max})$ *with* $l_{max}$ *fixed, i. e. only* $l_{max}$ *search directions* $q_{k-i,k}$ *are used for the calculation of the new iterate,*

- **combined**, *if* $l_k = \min((k-1) \mod l_{res} + 1\ ,\ l_{max})$, *i. e. if the truncated method is restarted.*

This definition covers all Krylov subspace methods[1] like `CG`, generalized `CG` as well as error-minimizing methods like `CGNE`, and classical iterative methods like `SOR` and Gauss-Seidel (see [42], [80]).
Any particular iterative method is characterized by the choice of the search directions $q_{k-i,k}$, by $l_k$ and by the orthogonalization matrices $Z_k$.
Despite of the generality of definition 3.6, some particular convergence estimates are possible. The following approximation theorem was given by Weiss in [80] for real linear systems. We augmented it by an analogous statement for complex linear systems.

**Theorem 3.7** (Approximation Theorem for Orthogonalization Methods, [80], p. 32)
*Let* $x_k$ *denote the iterates, let* $q_{k-i,k}$ *with* $i = 1, \ldots, l_k$ *denote the search directions and let* $Z_k$ *denote the orthogonalization matrices of an orthogonalization method applied to the linear system* $Ax = b$ *from* (3.1). *Let relation* (3.19) *have the form*

$$x_k \in \tilde{x}_k + \operatorname{span}\left(q_{k-l_k,k}, \ldots, q_{k-1,k}\right),$$

*and let* $\tilde{r}_k = A\tilde{x}_k - b$ *and* $\tilde{e}_k = \tilde{x}_k - x$.

*i) Let the matrix* $Z_k A^{-1} \in I\!R^{n\times n}$ *be positive real. Let* $Z_k A^{-1} = M + R$, *where* $M$ *is the symmetric and* $R$ *is the skew-symmetric part of* $Z_k A^{-1}$. *Let* $\rho(R)$ *denote the spectral radius of* $R$ *and* $\mu_m(M)$ *denote the minimal eigenvalue of* $M$.
*Then for all orthogonalization methods the following inequalities hold:*

$$\left\|r_k\right\|_{Z_k A^{-1}} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{\eta_1,\ldots,\eta_{l_k}} \left\|\textstyle\sum_{i=1}^{l_k} \eta_i A q_{k-i,k} + \tilde{r}_k\right\|_{Z_k A^{-1}} \tag{3.21}$$

*and*

$$\left\|e_k\right\|_{A^T Z_k} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{\eta_1,\ldots,\eta_{l_k}} \left\|\textstyle\sum_{i=1}^{l_k} \eta_i q_{k-i,k} + \tilde{e}_k\right\|_{A^T Z_k}, \tag{3.22}$$

*with* $\eta_i \in I\!R$ *for* $i = 1, \ldots, l_k$.

[1] see section 3.5

*ii) Let the matrix $Z_k A^{-1} \in I\!\!K^{n \times n}$ be Hermitian positive definite. Then the following equations hold for all orthogonalization methods:*

$$\left\| r_k \right\|_{Z_k A^{-1}} = \min_{\eta_1, \ldots, \eta_{l_k}} \left\| \sum_{i=1}^{l_k} \eta_i A q_{k-i,k} + \tilde{r}_k \right\|_{Z_k A^{-1}} \tag{3.23}$$

*and*

$$\left\| e_k \right\|_{A^H Z_k} = \min_{\eta_1, \ldots, \eta_{l_k}} \left\| \sum_{i=1}^{l_k} \eta_i q_{k-i,k} + \tilde{e}_k \right\|_{A^H Z_k}, \tag{3.24}$$

*with $\eta_i \in I\!\!K$ for $i = 1, \ldots, l_k$.*

**Proof.**

For the proof of assertion $i)$ and assertion $ii)$ with $I\!\!K = I\!\!R$, we refer to [80]. We give the proof for assertion $ii)$ with $I\!\!K = \mathbb{C}$, which proceeds analogously to the proof for the real case.
We prove assertion (3.23):

First, we show that the equation

$$r_k^H Z_k A^{-1} r_k = r_k^H Z_k A^{-1} \left( \sum_{i=1}^{l_k} \eta_i A q_{k-i,k} + \tilde{r}_k \right) \tag{3.25}$$

holds for arbitrary $\eta_i \in \mathbb{C}$ with $i = 1 \ldots, l_k$:

By (3.19), we know that

$$r_k = \sum_{i=1}^{l_k} \gamma_{i,k} A q_{k-i,k} + \tilde{r}_k \tag{3.26}$$

with some complex numbers $\gamma_{i,k}$. Now we have

$$\begin{aligned} r_k^H Z_k A^{-1} r_k &= r_k^H Z_k A^{-1} \left( \sum_{i=1}^{l_k} \gamma_{i,k} A q_{k-i,k} + \tilde{r}_k \right) \\ &\overset{(3.20)}{=} r_k^H Z_k A^{-1} \left( \sum_{i=1}^{l_k} \eta_i A q_{k-i,k} + \tilde{r}_k \right), \end{aligned}$$

where $\eta_i$ are arbitrary complex numbers for $i = 1, \ldots, n$ and the proof of equation (3.25) is complete.

We define the vector $\hat{r}_k := \sum_{i=1}^{l_k} \eta_i A q_{k-i,k} + \tilde{r}_k \in \mathbb{C}^n$ with $\eta_i \in \mathbb{C}$ for $i = 1, \ldots, j_k$, such that

$$\left\|\hat{r}_k\right\|_{Z_k A^{-1}} = \min_{\eta_1,\ldots,\eta_{l_k}} \left\|\sum_{i=1}^{l_k} \eta_i A q_{k-i,k} + \tilde{r}_k\right\|_{Z_k A^{-1}} .$$

By equation (3.25), we can write

$$\begin{aligned} \left\|r_k\right\|^2_{Z_k A^{-1}} &= r_k^H Z_k A^{-1} \hat{r}_k \\ &\leq \left\|r_k\right\|_{Z_k A^{-1}} \left\|\hat{r}_k\right\|_{Z_k A^{-1}} \end{aligned} \tag{3.27}$$

by the Cauchy-Schwarz inequality. Division by $\left\|r_k\right\|_{Z_k A^{-1}}$ gives us

$$\left\|r_k\right\|_{Z_k A^{-1}} \leq \left\|\hat{r}_k\right\|_{Z_k A^{-1}} . \tag{3.28}$$

The equality follows by noting that $r_k$ has the same representation as $\hat{r}_k$ by (3.26), and $\hat{r}_k$ is by definition the minimizer of the $Z_k A^{-1}$-norm of all such vectors.

Equation (3.24) follows with $Ae_k = r_k$.

$\diamondsuit$

Theorem 3.7 gives both a qualitative and a quantitative statement on the convergence of orthogonalization methods. If the root-expression in inequality (3.21) is relatively small, the convergence speed of the iterative solve is settled by the norm-expression in the right-hand side of (3.21), and hence the quantitative speed of convergence is determined by the search directions $q_{k-i,k}$ of the particular orthogonalization method. The quality of the approximation of the iterates $x_k$ to the exact solution depends on the vector norm induced by the matrices $Z_k A^{-1}$.
We remark that for particular projection methods more sophisticated estimates may be possible. A major attainment of the theorem 3.7 is its generality. Since the assertions of theorem 3.7 apply for general projection methods, a unifying theoretical framework is established for the large variety of known orthogonalization methods. For an ample interpretation of theorem 3.7 and for a circumstantial discourse on the classification of iterative methods in terms of orthogonalization methods, we refer to [80].
By theorem 3.7, the norms of the residuals decrease monotonously in the vector norm induced by the matrices $Z_k A^{-1}$. But it depends on the particular orthogonalization method, or more precisely on the particular choice of the orthogonalization matrices $Z_k$, whether or not the residual norms, which are applied as the stopping-criterion

for the iteration in practice, can be calculated in the vector norm induced by the matrices $Z_k A^{-1}$. If this is not possible, the Euclidian norms of the residuals can be applied instead as a stopping-criterion for the iteration. Since in this case the norms of the residuals may oscillate, using them as the stopping-criterion for the iteration may be somewhat unreliable. As a remedy for this problem the smoothing algorithm presented in section 3.4 can be applied. By this algorithm two supplemental vectors – the smoothed iterates and residuals – are calculated in each iteration step of the orthogonalization method. The Euclidian norms of the smoothed residuals decrease monotonously and are thus well suited for judging the convergence of the iterative solve.

We consider the statement of theorem 3.7 applied to the preconditioned linear system (3.6):

**Corollary 3.8** (Approximation Theorem for Preconditioned Linear Systems, [80], pp. 150–151)
*We consider the linear system $Ax = b$ from (3.1) and the corresponding preconditioned linear system $P_L A P_R y = P_L b$, $P_R y = x$ from (3.6). Let $y_k$ denote the iterates, let $\bar{q}_{k-i,k}$ with $i = 1, \ldots, l_k$ denote the search directions and let $Z_k$ denote the orthogonalization matrices of an orthogonalization method applied to the preconditioned linear system. Let relation (3.19) have the form*

$$y_k \in \tilde{y}_k + \operatorname{span}\left(\bar{q}_{k-l_k,k}, \ldots, \bar{q}_{k-1,k}\right),$$

*and let $\tilde{r}_k = AP_R\tilde{y}_k - b$, and $\tilde{e}_k = P_R\tilde{y}_k - x$.*

*i) Let the matrix $Z_k(P_L A P_R)^{-1} \in \mathbb{R}^{n\times n}$ be positive real, let $M$ denote the symmetric part and let $R$ denote the skew-symmetric part of $Z_k(P_L A P_R)^{-1}$. Let $\rho(R)$ denote the spectral radius of $R$ and $\mu_m(M)$ denote the minimal eigenvalue of $M$. Let $r_k$ and $e_k$ denote the residuals and errors of the original system (3.1) obtained from equations (3.10) and (3.11).*
*Then for all orthogonalization methods the following inequalities hold:*

$$\left\|r_k\right\|_{P_L^T Z_k (AP_R)^{-1}} \leq \sqrt{1+\frac{\rho^2(R)}{\mu_m^2(M)}} \min_{\eta_1,\ldots,\eta_{l_k}} \left\|\sum_{i=1}^{l_k} \eta_i AP_R \bar{q}_{k-i,k} + \tilde{r}_k\right\|_{P_L^T Z_k (AP_R)^{-1}} \tag{3.29}$$

*and*

$$\left\|e_k\right\|_{A^T P_L^T Z_k P_R^{-1}} \leq \sqrt{1+\frac{\rho^2(R)}{\mu_m^2(M)}} \min_{\eta_1,\ldots,\eta_{l_k}} \left\|\sum_{i=1}^{l_k} \eta_i P_R \bar{q}_{k-i,k} + \tilde{e}_k\right\|_{A^T P_L^T Z_k P_R^{-1}} \tag{3.30}$$

*with* $\eta_i \in \mathbb{R}$ *for* $i = 1, \ldots, l_k$.

*ii) Let the matrix* $Z_k(P_L A P_R)^{-1} \in \mathbb{K}^{n\times n}$ *be Hermitian positive definite. Then the equations*

$$\left\|r_k\right\|_{P_L^H Z_k (AP_R)^{-1}} = \min_{\eta_1,\ldots,\eta_{l_k}} \left\| \sum_{i=1}^{l_k} \eta_i A P_R \bar{q}_{k-i,k} + \tilde{r}_k \right\|_{P_L^H Z_k (AP_R)^{-1}} \tag{3.31}$$

*and*

$$\left\|e_k\right\|_{A^H P_L^H Z_k P_R^{-1}} = \min_{\eta_1,\ldots,\eta_{l_k}} \left\| \sum_{i=1}^{l_k} \eta_i P_R \bar{q}_{k-i,k} + \tilde{e}_k \right\|_{A^H P_L^H Z_k P_R^{-1}} \tag{3.32}$$

*hold with* $\eta_i \in \mathbb{K}$ *for* $i = 1, \ldots, l_k$.

**Proof.**

The proof for assertion $i)$ and for the real case of assertion $ii)$ of the corollary above is given in [80].
The proof of assertion $ii)$ for $\mathbb{K} = \mathbb{C}$ proceeds analogously:
By applying theorem 3.7 to the preconditioned linear system $P_L A P_R y = P_L b,\ P_R y = x$, we obtain

$$\left\|\bar{r}_k\right\|_{Z_k(P_L A P_R)^{-1}} = \min_{\eta_1,\ldots,\eta_{l_k}} \left\| \sum_{i=1}^{l_k} \eta_i P_L A P_R \bar{q}_{k-i,k} + P_L \tilde{r}_k \right\|_{Z_k(P_L A P_R)^{-1}}. \tag{3.33}$$

We consider the left-hand side of the above equation:

$$\begin{aligned} \left\|\bar{r}_k\right\|^2_{Z_k(P_L A P_R)^{-1}} &= r_k^H P_L^H Z_k (AP_R)^{-1} P_L^{-1} P_L r_k \\ &= r_k^H P_L^H Z_k (AP_R)^{-1} r_k \\ &= \left\|r_k\right\|^2_{P_L^H Z_k (AP_R)^{-1}}. \end{aligned}$$

We consider the norm expression in the right-hand side of equation (3.33):

$$\begin{aligned} &\left\| \sum_{i=1}^{l_k} \eta_i P_L A P_R \bar{q}_{k-i,k} + P_L \tilde{r}_k \right\|_{Z_k(P_L A P_R)^{-1}} \\ =&\left\| P_L \left( \sum_{i=1}^{l_k} \eta_i A P_R \bar{q}_{k-i,k} + \tilde{r}_k \right)) \right\|_{Z_k(P_L A P_R)^{-1}} \\ =&\left\| \sum_{i=1}^{l_k} \eta_i A P_R \bar{q}_{k-i,k} + \tilde{r}_k \right\|_{P_L^H Z_k (AP_R)^{-1}}. \end{aligned}$$

Assertion (3.32) follows analogously with $\bar{r}_k = P_L A P_R \bar{e}_k$.

$\diamondsuit$

Corollary 3.8 provides us with a qualitative and quantitative statement on the convergence of the preconditioned iterative solve in terms of the residuals and errors of the original linear system. Note that for the orthogonalization method applied to the preconditioned linear system not only different search directions $\bar{q}_{k-i,k}$ are used in contrast to the unpreconditioned linear system, but that those search directions are multiplied with the right-hand side preconditioning matrix $P_R$. Hence, the quantitative speed of convergence is altered by preconditioning. As for the qualitative speed of convergence, the residuals of the original linear system are minimized in a different induced matrix-norm in contrast to the unpreconditioned case.

## 3.4 Residual-Minimizing Smoothing

The following problem of iterative methods is known from practice: the norm of the residuals may heavily oscillate along the iteration process. This makes stopping criteria based on monitoring the residual norms unreliable. The purpose of the algorithm resulting from the following theoretical considerations is to generate an auxiliary sequence of vectors – the smoothed residuals – along the original residuals of the iteration, which decreases monotonously in some norm. This norm of the smoothed residuals is applied as stopping criterion for the iteration.

The smoothing algorithm described in the following was originally introduced by Schönauer ([64], [66], [67]) with the purpose of getting a function of the residuals which ensures a monotonous decrease of the Euclidian norm of the residuals for generalized conjugate gradient methods. However, the resulting smoothing algorithm can be applied to any iterative method.

The following definition of the smoothed sequence for the case $\mathbb{K} = \mathbb{R}$ is found in [80]. We consider both real and complex linear systems, i. e. $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$:

**Definition 3.9** (Smoothed Sequence, [80], p. 131)
*Let the vectors $x_k \in \mathbb{K}^n$ be the iterates and the vectors $r_k \in \mathbb{K}^n$ be the residuals of an iterative method for $k \in \mathbb{N}$. Then we call the sequence*

$$\begin{aligned} s_0 :=& r_0, z_0 := x_0, \\ s_k :=& s_{k-1} + \gamma_k (r_k - s_{k-1}), \\ z_k :=& z_{k-1} + \gamma_k (x_k - z_{k-1}), \end{aligned} \tag{3.34}$$

*with $s_k, z_k \in \mathbb{K}^n$ and $\gamma_k \in \mathbb{K}$, the corresponding* **smoothed sequence**. *The vectors $s_k$ are called* **smoothed residuals** *and the vectors $z_k$ are called* **smoothed iterates**.

Since the introduction of smoothing, many variants of Schönauer's original approach have been suggested which differ as for the determination of the coefficients $\gamma_k$. We

confine ourself to present Schönauer's Residual-Minimizing smoothing approach, and refer for an ample discussion of smoothing to [80].
The following definition of the Residual-Minimizing smoothed sequence for the case $\mathbb{K} = \mathbb{R}$ is found in [80]. We consider both real and complex linear systems, i. e. $\mathbb{K} \in \{\mathbb{R},\mathbb{C}\}$:

**Definition 3.10** (Residual-Minimizing Smoothing, [80], p.132)
*Let the vectors $x_k \in \mathbb{K}^n$ be the iterates and the vectors $r_k \in \mathbb{K}^n$ be the residuals of an iterative method for $k \in \mathbb{N}$. Further, let $z_k$ and $s_k$ denote the smoothed iterates and residuals from definition* 3.9. *Then the smoothed sequence obtained with*

$$\gamma_k := -\frac{\left(r_k - s_{k-1}\right)^H s_{k-1}}{\left\|r_k - s_{k-1}\right\|_2^2}, \tag{3.35}$$

*for $k \geq 1$, is called* **Residual-Minimizing smoothed sequence**, *and the smoothed iterates $z_k$ and residuals $s_k$ are called* **Residual-Minimizing smoothed iterates** *and* **residuals**, *respectively.*

The following lemma summarizes some properties of the Residual-Minimizing smoothed residuals.

**Lemma 3.11** (Properties of the Residual-Minimizing Smoothed Sequence, [64], pp. 261–262)
*Let the vectors $x_k \in \mathbb{K}^n$ be the iterates and the vectors $r_k \in \mathbb{K}^n$ be the residuals of an iterative method for $k \in \mathbb{N}$. Furthermore, let $z_k$ and $s_k$ denote the Residual-Minimizing smoothed iterates and residuals from definition* 3.10. *Then the statements*

$$\left\|s_k\right\|_2 = \min_{\tau \in \mathbb{K}} \left\|s_{k-1} + \tau \left(r_k - s_{k-1}\right)\right\|_2, \tag{3.36}$$

$$\left\|s_k\right\|_2 \leq \left\|s_{k-1}\right\|_2 \tag{3.37}$$

*and*

$$\left\|s_k\right\|_2 \leq \min_{i=1,\ldots,k} \left\|r_i\right\|_2 \tag{3.38}$$

*hold for $k \geq 1$.*

**Proof.**

We prove the assertions for $\mathbb{K} = \mathbb{C}$ only. The proof for the case $\mathbb{K} = \mathbb{R}$ is given in [64], pp. 261–262.
First, we verify assertion (3.36):
By (3.34) , it suffices to show that the number $\gamma_k \in \mathbb{C}$ defined in (3.35) is the minimizer of

$$\min_{\tau \in \mathbb{C}} \left\| s_{k-1} + \tau \left( r_k - s_{k-1} \right) \right\|_2^2 .$$

We consider in which location the function $f : \mathbb{C} \to \mathbb{R}$ defined by

$$f(\tau) := \left\| s_{k-1} + \tau \left( r_k - s_{k-1} \right) \right\|_2^2$$

attains its absolute minimum. With elementary algebraic conversion we obtain

$$f(\tau) = \left( \tau_1^2 + \tau_2^2 \right) \left\| r_k - s_{k-1} \right\|_2^2 + 2 \left( \tau_1 \zeta_1 - \tau_2 \zeta_2 \right) + \left\| s_{k-1} \right\|_2^2 ,$$

with $\tau := \tau_1 + i\tau_2$ and $\zeta_1 + i\zeta_2 := s_{k-1}^H \left( r_k - s_{k-1} \right)$.
In order to find a minimum of the function $f$ we consider where its first derivative vanishes:

$$\begin{aligned} f'(\tau) &= \left( 2\tau_1 \left\| r_k - s_{k-1} \right\|_2^2 + 2\zeta_1, 2\tau_2 \left\| r_k - s_{k-1} \right\|_2^2 - 2\zeta_2 \right) \\ &= (0, 0) \end{aligned}$$

$$\begin{aligned} \Longleftrightarrow \quad \tau \quad &= \quad -\frac{\zeta_1}{\left\| r_k - s_{k-1} \right\|_2^2} + i \frac{\zeta_2}{\left\| r_k - s_{k-1} \right\|_2^2} \\ &= \quad -\frac{\left( r_k - s_{k-1} \right)^H s_{k-1}}{\left\| r_k - s_{k-1} \right\|_2^2} \\ &\overset{(3.35)}{=} \quad \gamma_k , \end{aligned}$$

and the proof of assertion (3.36) is complete.
Assertions (3.37) and (3.38) follow directly from (3.36) with $\tau := 0$ and $\tau := 1$, respectively. $\diamondsuit$

Equations (3.37) and (3.38) state that the smoothed sequence $s_k$ decreases monotonously in the Euclidian norm. Thus, by monitoring the norms of the Residual-Minimizing smoothed residuals a reliable stopping-criterion for the iterative solve is at hand.

**The Residual-Minimizing Smoothing Algorithm**

We state the Residual-Minimizing smoothing algorithm for the Euclidian norm:

Let $x_k$ be the iterate and $r_k$ be the corresponding residual of an iterative method. Let

$$z_0 := x_0$$
$$s_0 := r_0$$

Calculate for $k \geq 1$:

$$\gamma_k := \frac{\left(r_k - s_{k-1}\right)^H s_{k-1}}{\left\|r_k - s_{k-1}\right\|_2^2}$$
$$z_k := z_{k-1} + \gamma_k\left(x_k - z_{k-1}\right)$$
$$s_k := s_{k-1} + \gamma_k\left(r_k - s_{k-1}\right)$$

**Algorithm 1:** `Residual-Minimizing Smoothing for the Euclidian Norm`

In figure 1, we give a plot of the convergence of a Krylov subspace method (namely the `BiCGstab` method, see section 3.5.3) applied to a linear system $Ax = b$ (with the coefficient matrix orsirr2, see section 7.2, and a random right-hand side vector $b$). The horizontal axes of the coordinate systems for the residuals and for the errors denote the iteration steps of the iterative solver. The vertical axis for the residuals denotes the decadic logarithm of the relative residuals, i.e. of the Euclidian norm of the residual in each iteration step divided by the Euclidian norm of the very first residual $r_0 := Ax_0 - b$, where $x_0$ denotes the initial guess for desired solution of the linear system. Analogously, the vertical axis for the errors denotes the decadic logarithm of the relative errors.

Note how the oscillations in the residual norms are canceled out be applying the Residual-Minimizing smoothing technique.

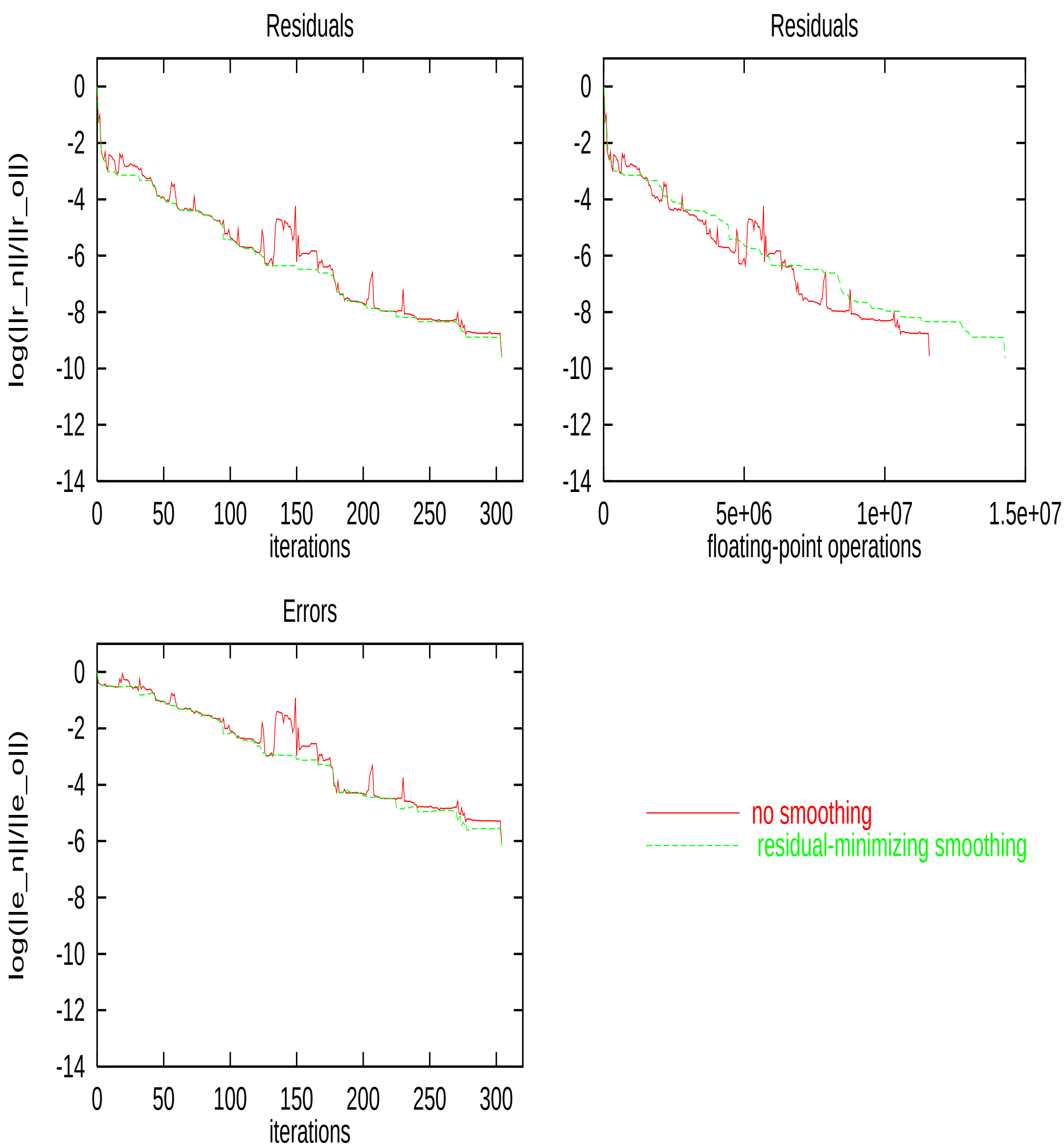


Figure 1: An example for `Residual-Minimizing Smoothing`

In figure 1, the effect of canceling out the peaks along the decrease of the residuals of the `BiCGstab` iteration can be observed. Since the matrix chosen for comparing the Residual-Minimizing smoothed iteration to the original iteration (this is orsirr2, see section 7.2) is relatively small and very sparse, the computational cost caused by Residual-Minimizing smoothing is quite high for this example. For larger and less sparse linear systems this additional computational cost becomes neglectable compared to the cost involved with the two matrix-vector multiplications in each iteration step of the `BiCGstab` solver.

**The Computational Complexity of the `Residual-Minimizing Smoothing` Algorithm**

Suppose that the $k$-th iterate $x_k \in I\!R^n$ and the corresponding residual $r_k \in I\!R^n$ are calculated by some iterative solver for the linear system $Ax = b$ with $A \in I\!R^{n\times n}$ non-singular and $x,\ b \in I\!R^n$. Then, for the calculation of the scalar $\gamma_k$ the vector $r_k - s_{k-1}$, according to definition 3.10, and its Euclidian norm must be computed. Furthermore, the dot-product $s_{k-1}^H(r_k - s_{k-1})$ has to be calculated. Those operations require at most $5n$ floating-point operations. For the calculation of the $k$-th terms $z_k$ and $s_k$ of the Residual-Minimizing smoothed iterates and residuals, the triadic operations $z_k = z_{k-1} + \gamma_k(r_k - s_{k-1})$ and $s_k = s_{k-1} + \gamma_k(r_k - s_{k-1})$ are necessary. Those operations require at most $4n$ floating point operations.
Note that in practice the number of floating-point operations may be substantially less, if the involved vectors are sparse.
The smoothing algorithm presented in this section is a useful tool for controlling the convergence of iterative methods. For a detailed investigation on the effect of smoothing on iterative solvers, we refer to [67].

## 3.5 Krylov Subspace Methods

In this section, we briefly describe Krylov subspace methods, a subclass of orthogonalization methods, which contains many of the state of the art iterative solvers. Krylov subspace methods are particularly interesting for the implementation on vector and parallel computers, since their computational complexity is dominated by matrix-vector multiplications, which are efficiently vectorizable and parallelizable. Furthermore, Krylov subspace methods applied with explicit preconditioning are very interesting for the implementation on vector and parallel computers, since the computational cost of applying them to preconditioned linear systems only originates from additional matrix-vector multiplications. For an ample survey on Krylov subspace methods, we refer to [62] and [80].

**Definition 3.12** (Krylov Subspace Methods)
*We use the nomenclature of definition* 3.6. *Let $S$ be a non-singular matrix in $I\!K^{n\times n}$, and $w$ be a vector in $I\!K^n$.* **Krylov subspace methods** *are orthogonalization methods with*

$$q_{k-i,k} \in K_{k-i}(S, w) = \mathrm{span}(w, Sw, S^2w, \ldots, S^{k-i}w) \tag{3.39}$$

*and*

$$\tilde{x}_k = x_{k-l_k}, \tag{3.40}$$

*where $w$ is a vector in $\mathbb{K}^n$. The linear space $K_t(S,w)$ is called $t$-th* **Krylov subspace** *of $S$ applied to the vector $w$.*

The following lemma states a property of Krylov subspace methods that makes them appealing for the implementation on vector and parallel computers: the computational complexity of Krylov subspace methods is dominated by matrix-vector multiplications. For the derivation and the proof of the assertions, we refer to [80].

**Lemma 3.13** ([80], p. 43)
*Let $x_k$, $r_k \in \mathbb{K}^n$ denote the iterates and the residuals of a Krylov subspace method applied to the linear system $Ax = b$ from (3.1), and let the search directions $q_{k-i,k} \in \mathbb{K}^n$ be elements of the Krylov subspace $K_{k-i}(S,w)$, with $S \in \mathbb{K}^{n \times n}$ non-singular. Then the iterates $x_k$ and the residuals $r_k$ have the following representation:*

$$x_k = \sum_{i=1}^{k} \beta_{i,k} S^{i-1} w + x_0 \tag{3.41}$$

*and*

$$r_k = \sum_{i=1}^{k} \beta_{i,k} A S^{i-1} w + r_0, \tag{3.42}$$

*with $\beta_{i,k} \in \mathbb{K}$ for $i = 1, \ldots, k$ and $k \geq 1$.*
*If $S \neq A$, two matrix-vector products must be computed in each iteration step of a Krylov subspace method: one involving the matrix $S$ in order to determine the search direction $q_{k-1,k}$ and another involving the matrix $A$ in order to compute the residual $r_k$. If $S = A$, in general one matrix-vector multiplication per iteration step is sufficient.*

The above lemma states that the computational complexity of Krylov subspace methods applied for solving the original linear system $Ax = b$ from (3.1) is dominated by matrix-vector multiplications. Since those operations are optimally vectorizable and parallelizable, those can be implemented most efficiently on today's supercomputers.

**Short Recurrences**

The representation of the iterates in (3.41) and of the residuals in (3.42) involves all previous iterates and residuals. Due to memory and CPU-time restrictions, in practice not all those vectors can be stored in computer memory during the entire iteration process and be involved in the calculation of the new iterates and residuals. However, for some Krylov subspace methods this is not necessary, since their iterates and residuals can be calculated with a short recurrence, i.e. by a calculation involving only two or three previous iterates and residuals. Methods that have no short recurrence provided by theory are truncated in practice, i.e. only a specified number of previous iterates and residuals is taken into account for calculating the new iterates and residuals. Although this introduces an error into the iteration process, truncated methods are very efficient for linear systems with certain properties. A detailed discussion on that matter is given by Weiss in [80], pp. 59–62.

**Preconditioned Krylov Subspace Methods**

We consider lemma 3.13 for Krylov subspace methods applied to the preconditioned linear system $P_L A P_R y = P_L b,\ P_R y = x$ from (3.6). The iterates $y_k$ and the residuals $\bar{r}_k$ from (3.10) of a Krylov subspace method applied to the preconditioned linear system $P_L A P_R y = P_L b,\ P_R y = x$ can be written – by lemma 3.13 – as

$$y_k = \sum_{i=1}^{k} \beta_{i,k} S^{i-1} w + y_0 \tag{3.43}$$

and

$$\bar{r}_k = \sum_{i=1}^{k} \beta_{i,k} P_L \left( A \left( P_R \left( S^{i-1} w \right) \right) \right) + \bar{r}_0, \tag{3.44}$$

with $\beta_{i,k} \in I\!\!K$ for $i = 1, \ldots, k$ and $k \geq 1$, where the search directions are elements of the Krylov subspace $K_{k-i}(S, w)$ with $w \in I\!\!K^n$. Thus, each iterate $y_k$ of the Krylov subspace method applied to the preconditioned linear system requires the matrix-vector product $v_1 := S^{k-1} w$. Additionally, each residual of the preconditioned Krylov subspace iteration requires the matrix-vector products $v_2 := P_R v_1$, $v_3 := A v_2$ and $P_L v_3$. Hence, for Krylov subspace methods it is sufficient to know the coefficient matrix $P_L A P_R$ in the form of matrix-vector multiples, i.e. it is not necessary to calculate the matrix-matrix product $P_L A P_R$ explicitly.

The iterative solvers considered in our numerical tests (`CG`, `PRES20`, `BiCGstab` and `ATPRES`, see sections 3.5.1–3.5.4) belong to the family of conjugate Krylov subspace methods. Conjugate Krylov subspace methods are Krylov subspace methods with

$$S = A$$

and

$$w = r_0$$

in (3.41) and (3.42), respectively. Thus, the iterates $y_k$ and the residuals $\bar{r}_k$ from (3.10) of a conjugate Krylov subspace method applied to the preconditioned linear system $P_L A P_R y{=}P_L b$, $P_R y = x$ have by lemma 3.13 the representation

$$y_k = \sum_{i=1}^{k} \beta_{i,k} \left(P_L A P_R\right)^{i-1} \bar{r}_0 + y_0 \tag{3.45}$$

and

$$\bar{r}_k = \sum_{i=1}^{k} \beta_{i,k} \left(P_L A P_R\right)^{i} \bar{r}_0 + \bar{r}_0, \tag{3.46}$$

where $\beta_{i,k} \in I\!\!K$ for $i = 1, \ldots, k$ and $k \geq 1$. The search directions $q_{k-i,k}$ from (3.39) of conjugate Krylov subspace methods are elements of the Krylov subspaces $K_{k-i}\left(P_L A P_R, \bar{r}_0\right)$. By equations (3.45) and (3.46), each iteration step of a conjugate Krylov subspace method applied to the preconditioned linear system $P_L A P_R y{=}P_L b$, $P_R y = x$ involves three matrix-vector multiplications. With the vectors $v_l \in I\!\!K^n$ for $l = 1, \ldots, 4$ and with $v_1 := \left(P_L A P_R\right)^{k-1} \bar{r}_0$ these matrix-vector multiplications are $v_2 := P_R v_1$, $v_3 := A v_2$ and $v_4 := P_L v_3$.

**Generalized CG-methods**

In the following, we describe the four Krylov subspace methods for real linear systems, i.e. for linear systems with coefficient matrices $A \in I\!\!R^{n \times n}$, which we considered in our numerical tests, in more detail. Further, we give the formulation of those methods for the preconditioned linear system $P_L A P_R y = P_L b$, $P_R y = x$ from (3.6) in a pseudo-code.

Our choice of the Krylov subspace methods considered in our numerical tests on preconditioning linear systems is representative in the following sense:

If the coefficient matrix $A$ of the linear system in question is symmetric positive definite, the CG method (described in section 3.5.1, see e.g. [37], [70]) can be applied to solve the preconditioned linear system. The iterates $x_k$ of the CG method can be calculated with a short recurrence, which is equivalent to minimizing a certain matrix functional (see (3.47) in section 3.5.1). If the coefficient matrix $A$ is not symmetric positive definite, then CG can still be applied, but convergence is not guaranteed, or may be slow, and additionally, the short recurrence is not equivalent to minimizing the mentioned matrix functional (see (3.47) in section 3.5.1).

During the last decades, a lot of generalized CG methods have been proposed for solving linear systems with an unsymmetric coefficient matrix. There are three main branches of generalized CG methods:

**Truncated, restarted** Those methods are obtained by ignoring the non-existence of a short recurrence. The iterates and residuals are calculated by

$$x_k = \sum_{i=k-t}^{k} \beta_{i,k} S^{i-1} w + x_0,$$

$$r_k = \sum_{i=k-t}^{k} \beta_{i,k} A S^{i-1} w + r_0,$$

i.e. only the $t$ preceeding iterates are involved in calculating the new iterates and residuals. From practical experiences with real linear systems and coefficient matrices close to normal, it is known that the resulting methods converge quickly, if the eigenvalues of the coefficient matrix are clustered in the positive halfplane close to the real axis, but such methods are not very robust.

For our numerical tests, we chose the `PRES20` method introduced by Schönauer in [64] as an representant for this branch of generalized `CG` methods. For this method, the parameter $t$ is set to five, and a restart, i.e. the explicit calculation $r_k := Ax_k - b$, is made every twenty iterations.

`BCG` **Approach** A variety of methods originate from considering the "doubled" linear system

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} x^* \\ x \end{pmatrix} = \begin{pmatrix} b \\ b^* \end{pmatrix}$$

instead of the original linear system $Ax = b$. The "doubled" linear system is symmetric, but in general not positive definite. The resulting methods can be implemented with a short recurrence, but the residuals are minimized in a "complicated" induced matrix norm. Numerical experiments with real linear systems and coefficient matrices close to normal indicate that the resulting methods work well, if the eigenvalues are scattered in the positive halfplane.

For our numerical tests, we selected the `BiCGstab`-method introduced by van der Vorst in [78] as a proxy for this kind of generalized `CG`-methods.

**Normal equations** Those methods are based on considering the normal equations

$$A^T A x = A^T b$$

instead of the original linear system $Ax = b$. The coefficient matrix $A^T A$ of the normal equations is symmetric positive definite, and thus the `CG`-method can be applied. Hence, methods resulting from this approach can be implemented with a short recurrence. From extensive numerical tests with real valued coefficient matrices close to normal (see e.g. [80], pp. 116–130), it is known that methods of this origin are very robust, i.e. they converge, even if the eigenvalues of the coefficient matrix are scattered all across the complex plane, but, since the condition number of the coefficient matrix of the normal equations $\kappa\left(A^T A\right)$ equals $\left(\kappa\left(A\right)\right)^2$, the square of the condition number of the original coefficient matrix $A$, those methods tend to converge relatively slow.

For our numerical tests, we chose the `ATPRES`-method introduced by Schönauer in [68].

### 3.5.1 The `CG`-method

The `CG`-method was introduced by Hestenes and Stiefel in [37]. An easy to read introduction is given in [70]. The `CG`-method can be implemented with a short recurrence. The residuals $r_k$ are orthogonal to all preceeding residuals $r_1, \ldots r_{k-1}$, and, equivalently, the iterates are calculated such that the functional

$$F(x) := \frac{1}{2}\left(Ax - b\right)^T A^{-1}\left(Ax - b\right) \tag{3.47}$$

is minimized, where $A$ is an element of $\mathbb{R}^{n \times n}$.

In algorithm 2, we give a pseudo-code formulation of the `CG`-method with Residual-Minimizing smoothing applied to the preconditioned linear system

$$P_L A P_R y = P_L b,\ P_R y = x$$

from (3.6).

Choose an initial guess $y_0$, set $\tilde{y}_0 := y_0$, $s := P_R y_0$, $t := As$, $\tilde{r}_0 := r_0 := P_L t - P_L b$ and calculate for $k \geq 1$:

$$\begin{aligned}
s &:= P_R \tilde{r}_{k-1} \\
t &:= As \\
u &:= P_L t \\
\gamma_k &:= -\frac{\tilde{r}_{k-1}^T \tilde{r}_{k-1}}{\tilde{r}_{k-1}^T u} \\
\tilde{y}_k &:= \tilde{y}_{k-1} + \gamma_k \tilde{r}_{k-1} \\
\tilde{r}_k &:= \tilde{r}_{k-1} + \gamma_k u \\
\alpha_k &:= -\frac{r_{k-1}^T (\tilde{r}_k - r_{k-1})}{(\tilde{r}_k - r_{k-1})^T (\tilde{r}_k - r_{k-1})} \\
r_k &:= r_{k-1} + \alpha_k (\tilde{r}_k - r_{k-1}) \\
y_k &:= y_{k-1} + \alpha_k (\tilde{y}_k - y_{k-1})
\end{aligned}$$

**Algorithm 2:** `CG + Residual-Minimizing Smoothing`

In algorithm 2, the original iterates and residuals are denoted by $\tilde{y}_k$ and $\tilde{r}_k$. The corresponding Residual-Minimizing smoothed iterates and residuals are $y_k$ and $r_k$. Per iteration step, three matrix-vector multiplications are made, namely $s := P_R \tilde{r}_{k-1}$, $t := As$ and $u := P_L t$ are calculated. If no preconditioning is applied, i.e. $P_L = P_R = I$, only one matrix-vector multiplication, namely $u := A\tilde{r}_{k-1}$, is necessary.

### 3.5.2 The `PRES20`-method

The name `PRES20` is an abbreviation of Pseudo-RESidual method 20. This method was introduced by Schönauer in 1987 (see [64]). The `PRES20`-method is a combined orthogonalization method with Residual-Minimizing smoothing (see section 3.4). Restarts, i.e. the explicit calculations $r_k := Az_k - b$, where $z_k$ denotes the Residual-Minimizing smoothed iterates, are made every twenty iterations and at most five search directions are used. The truncation- and the restart-parameter were optimized along extensive numerical tests. The orthogonalization matrix $Z_k$ is constantly the identity matrix. The search directions from definition 3.6 $q_{k-i,k}$ are elements of the Krylov subspace $K_{k-i}(A, r_0)$.

In algorithm 3, we give a pseudo-code formulation of the `PRES20`-method applied to the preconditioned linear system $P_L A P_R y = P_L b$, $P_R y = x$ from (3.6). The Residual-Minimizing smoothed iterates and residuals of the `PRES20` iteration are denoted by $y_k$ and $r_k$. The corresponding non-smoothed iterates and residuals are $\tilde{y}_k$ and $\tilde{r}_k$. Per iteration step, three matrix-vector multiplications, namely $s := P_R \tilde{r}_{k-1}$, $t := As$ and $u := P_L t$, are made. If no preconditioning is applied, i.e. $P_L = P_R =$

$I$, only one matrix-vector multiplication, namely $u := A\tilde{r}_{k-1}$, is necessary. The `PRES20` method is a very efficient iterative solver for matrices close to normal, if the eigenvalues of the matrices lie in the positive half-plane close to the real axis. For further details see e.g. [64], [80], or [81].

Choose an initial guess $y_0$, set $\tilde{y}_0 := y_0$, $s := P_R y_0$, $t := As$, $\tilde{r}_0 := r_0 := P_L t - P_L b$ and calculate for $k \geq 1$:

$$
\begin{aligned}
& s := P_R \tilde{r}_{k-1} \\
& t := As \\
& u := P_L t \\
& \alpha_{i,k} := -\frac{\tilde{r}_{k-i}^T u}{\tilde{r}_{k-i}^T \tilde{r}_{k-i}} \ \ for\ i = 1, \ldots, \min(k,5) \\
& \phi_k := \frac{1}{\sum_{i=1}^{\min(k,5)} \alpha_{i,k}} \\
& \tilde{y}_k := \phi_k \left( \tilde{r}_{k-1} + \sum_{i=1}^{\min(k,5)} \alpha_{i,k} \tilde{y}_{k-i} \right) \\
& \tilde{r}_k := \phi_k \left( u + \sum_{i=1}^{\min(k,5)} \alpha_{i,k} \tilde{r}_{k-i} \right) \\
& \gamma_k := -\frac{r_{k-1}^T (\tilde{r}_k - r_{k-1})}{(\tilde{r}_k - r_{k-1})^T (\tilde{r}_k - r_{k-1})} \\
& r_k := r_{k-1} + \gamma_k (\tilde{r}_k - r_{k-1}) \\
& y_k := y_{k-1} + \gamma_k (\tilde{y}_k - y_{k-1}) \\
& \text{If } \ (\operatorname{mod}(k, 20) == 0) \ \text{ Restart}
\end{aligned}
$$

**Algorithm 3:** `PRES20`

### 3.5.3 The `BiCGstab`-method

The `BiCGstab`-method was introduced by van der Vorst [78] in 1992. The name `BiCGstab` is a short form of Bi-Conjugate Gradients STABilized.
In the following, we give a brief sketch on the derivation of BiCG-like methods for $A \in \mathbb{R}^{n \times n}$ non-singular (detailed considerations on that matter are given e.g. in [78] and [80]).

All BiCG-like methods are essentially based on working with a "doubled" system

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} x^* \\ x \end{pmatrix} = \begin{pmatrix} b \\ b^* \end{pmatrix}, \tag{3.48}$$

instead of the original linear system $Ax = b$. The "doubled" linear system is symmetric, but not necessarily positive definite. The vector $b^* \in I\!R^n$ is arbitrary. The iterates $x_k$ and $x_k^* \in I\!R^n$ are determined by the following two formulae:

$$x_k = x_{k-1} + \sum_{i=1}^{k} \varphi_{i,k} r_{i-1}$$

and

$$x_k^* = x_{k-1}^* + \sum_{i=1}^{k} \varphi_{i,k} r_{i-1}^*,$$

where the residuals $r_k$ and $r_k^* \in I\!R^n$ are given by

$$r_k = Ax_k - b$$

and

$$r_k^* = A^T x_k^* - b^*.$$

The coefficients $\varphi_{i,k} \in I\!R$ are obtained from the biorthogonalities

$$r_k^T r_{k-i}^* = 0$$

and

$$\left(r_k^*\right)^T r_{k-i} = 0$$

for $i = 1, \ldots, k$. The orthogonality equations above can be considered as a weak formulation for the vanishing of the residuals for the exact solution.
The residuals can be represented by a matrix polynomial $\Pi_k\left(A\right)$ of the form

$$\Pi_k\left(A\right) := \sum_{i=1}^{k} \gamma_i A^i + I,$$

where $I$ denotes the $(n \times n)$-identity matrix. The polynomial notation of the residuals is then

$$r_k = \Pi_k\left(A\right) r_0 \tag{3.49}$$

and

$$r_k^* = \Pi_k\left(A^T\right) r_o^*,$$

where $r_o$ and $r_o^*$ denote the residuals of the initial guess $x_o$ and $x_o^*$. The basic iterative solver obtained by this approach is called `BCG`.
In [73], a variation of this approach, the `CGS` algorithm, was introduced by Sonneveld. Instead of considering the residuals in the form (3.49), residuals in the form

$$\tilde{r}_k := \Pi_k^2(A)\, r_0$$

and the corresponding errors

$$\tilde{e}_k := \Pi_k^2(A)\, e_0$$

are considered, i.e. instead of the polynomial $\Pi_k(A)$, the squared polynomial $\Pi_k^2(A)$ is considered. The advantage of this approach is, that if $\Pi_k(A)$ tends to zero, then faster convergence is obtained with the squared polynomial $\Pi_k^2(A)$. In practice, the convergence of `CGS` often is faster than the convergence of the original `BCG`. On the other hand, the convergence of `CGS` may be more erratic than `BCG`.
In order to obtain a more stable and smoother converging iterative solver than `CGS`, the `BiCGstab`-algorithm was proposed by van der Vorst in [78]. The idea of the `BiCGstab`-algorithm is to consider the matrix polynomial $\Pi_k(A)$ multiplied by a matrix polynomial $\Psi_k(A)$ instead of considering the squared matrix polynomial $\Pi_k^2(A)$ of the `CGS`-iteration, i.e. the iterates and the residuals of the `BiCGstab`-iteration have the representation

$$r_k := \Psi_k(A)\,\Pi_k(A)\, r_0$$

and the corresponding errors

$$e_k := \Psi_k(A)\,\Pi_k(A)\, e_0.$$

The matrix polynomial $\Psi_k(A)$ is recursively defined by

$$\Psi_k(A) := (\eta_k A + I)\,\Psi_{k-1}(A),$$

where the coefficient $\eta_k \in I\!R$ is calculated by the one-dimensional minimization

$$\left\|r_k\right\|_2 = \min_{\eta_k \in I\!K} \left\|(\eta_k A + I)\,\Psi_{k-1}(A)\,\Pi_k(A)\, r_o\right\|_2.$$

We give a pseudo-code formulation of the `BiCGstab`-method including Residual-Minimizing smoothing applied to the preconditioned linear system $P_L A P_R y = P_L b$, $P_R y = x$ from (3.6) in algorithm 4.

Choose $r_0^* \neq 0$ and an initial guess $y_0$, $l := P_R y_0$, $m := Al$, set $r_0 := P_L m - P_L b$, $z_0 := y_0$, $g_0 := r_0$, $\rho_0 := [r_0^*]^T r_0$, $p_0 := r_0$ and calculate for $k \geq 1$:

$$
\begin{aligned}
l &:= P_R p_{k-1} \\
m &:= Al \\
v &:= P_L m \\
\varphi_k &:= -\frac{\rho_{k-1}}{[r_0^*]^T v} \\
s &:= r_{k-1} + \varphi_k v \\
l &:= P_R s \\
m &:= Al \\
t &:= P_L m \\
\gamma_k &:= -\frac{s^T t}{\|t\|_2^2} \\
r_k &:= s + \gamma_k t \\
y_k &:= y_{k-1} + \varphi_k p_{k-1} + \gamma_k s \\
\rho_k &:= [r_0^*]^T r_k \\
\beta_k &:= \frac{\rho_k}{\rho_{k-1}} \frac{\varphi_k}{\gamma_k} \\
p_k &:= r_k + \beta_k (p_{k-1} + \gamma_k v) \\
\psi_k &:= \frac{g_{k-1}^T (r_k - g_{k-1})}{\|r_k - g_{k-1}\|_2^2} \\
z_k &:= z_{k-1} + \psi_k (y_k - z_{k-1}) \\
g_k &:= g_{k-1} + \psi_k (r_k - g_{k-1})
\end{aligned}
$$

**Algorithm 4:** `BiCGstab + Residual-Minimizing Smoothing`

In algorithm 4, the original iterates and residuals of the `BiCGstab` iteration are denoted by $y_k$ and $r_k$. The corresponding Residual-Minimizing smoothed iterates and residuals are $z_k$ and $g_k$. Per iteration step, six matrix-vector products are required; three for calculating $P_L A P_R p_{k-1}$, i.e. $l := P_R p_{k-1}$, $m := Al$ and $v := P_L m$, and three for $P_L A P_R s$, i.e. $l := P_R s$, $m := Al$ and $t := P_L m$. If no preconditioning is applied, only two matrix-vector multiplications involving the matrix $A$, namely $v := Ap_{k-1}$ and $t := As$, are necessary. Extensive numerical tests on real valued close to normal matrices suggest that this method performs well, if the eigenvalues of the matrix $A$ are spread across the positive half-plane. For results of numerical tests, see e.g. [29], [71], and [72].

The `BiCGstab`-method is not yet classified in terms of orthogonalization methods, because the orthogonalization matrices are unknown.

### 3.5.4 The ATPRES-method

The name `ATPRES` is an abbreviation for $A$ Transposed Pseudo RESidual. This method was introduced by Schönauer in [68] for real linear systems. It is based on the `CG`-method applied to the normal equations

$$A^T Ax = A^T b, \tag{3.50}$$

with Residual-Minimizing smoothing (see section 3.4, [65], [66], [80]). The orthogonalization matrix $Z_k$ is constantly the matrix $A$.
We give a pseudo-code formulation of the `ATPRES`-method applied to the preconditioned linear system $P_L A P_R y = P_L b,\ P_R y = x$ from (3.6) in algorithm 5.
In algorithm 5, the Residual-Minimizing smoothed iterates and residuals are $y_k$ and $r_k$. The corresponding non-smoothed iterates and residuals are $\tilde{y}_k$ and $\tilde{r}_k$. Per iteration step, six matrix-vector products are necessary: three for evaluating $(P_L A P_R)^T\, \tilde{r}_{k-1}$, i.e. $s := P_L^T \tilde{r}_{k-1}$, $t := A^T s$, $u := P_R^T t$, and three for $P_L A P_R u$, i.e. $v := P_R u$, $w := Av$ and $n := P_L w$. If no preconditioning is applied, only the two matrix-vector products $u := A^T \tilde{r}_{k-1}$ and $n := Au$ are necessary in each iteration step. Numerical experiments with the `ATPRES`-method on real linear systems with coefficient matrices not far from normal indicate that this method is the most robust of the four and can be applied for any distribution of the eigenvalues of the matrix $A$.

Choose an initial guess $y_0$, set $\tilde{y}_0 := y_0$, $s := P_R y_0$, $t := As$, $\tilde{r}_0 := r_0 := P_L t - P_L b$ and calculate for $k \geq 1$:

$$
\begin{aligned}
& s := P_L^T \tilde{r}_{k-1} \\
& t := A^T s \\
& u := P_R^T t \\
& v := P_R u \\
& w := Av \\
& n := P_L w \\
& \alpha_{i,k} := -\frac{\tilde{r}_{k-i}^T n}{\tilde{r}_{k-i}^T \tilde{r}_{k-i}} \text{ for } i = \min(k, 2), \\
& \phi_k := \frac{1}{\sum_{i=1}^{\min(k,2)} \alpha_{i,k}} \\
& \tilde{y}_k := \phi_k \left( u + \sum_{i=1}^{\min(k,2)} \alpha_{i,k} \tilde{y}_{k-i} \right) \\
& \tilde{r}_k := \phi_k \left( n + \sum_{i=1}^{\min(k,2)} \alpha_{i,k} \tilde{r}_{k-i} \right) \\
& \gamma_k := -\frac{r_{k-1}^T (\tilde{r}_k - r_{k-1})}{(\tilde{r}_k - r_{k-1})^T (\tilde{r}_k - r_{k-1})} \\
& r_k := r_{k-1} + \gamma_k (\tilde{r}_k - r_{k-1}) \\
& y_k := y_{k-1} + \gamma_k (\tilde{y}_k - y_{k-1})
\end{aligned}
$$

**Algorithm 5:** `ATPRES`

# 4 Projection Methods

In this chapter, we consider projection methods – a general class of methods for calculating sparse approximate inverses for non-singular real or complex matrices. We introduce algorithms for the calculation of approximate inverses by projection methods, and we derive estimates on the "distance" between the true and the approximate inverse calculated by projection methods.

Because of the general definition of projection methods, some known preconditioning approaches can be formulated in terms of projection methods. Thus, for those methods a unifying theoretical framework is supplied (see chapter 6).
Further, the theoretical framework of projection methods can be utilized as building kit for new preconditioning methods with both a theoretical framework and favorable algorithmic properties. In particular, the resulting preconditioning methods offer a large amount of parallelism, and thus are appealing for parallel computers (see chapter 6).

We begin section 4.1 with introducing some nomenclature for the pattern of vectors and matrices. We give the definition of projection methods and we derive an explicit representation of the corresponding approximate inverses. Furthermore, we give two pseudo-code algorithms for the calculation of approximate inverses with projection methods.
In section 4.2, we establish an approximation theorem that gives a statement on the "distance" between the approximate and the true inverse in form of a minimization property.

## 4.1 The General Concept

By projection methods, the non-zero entries of one-sided (either left-hand side or right-hand side) approximate inverses with fixed sparsity-patterns can be calculated. Because of CPU-time and memory restrictions in practice, the approximate inverse should be as sparse as possible. Thus, the choice of its pattern, i.e. the location of the non-zero entries in the approximate inverse, is very important.
We introduce some notation for the pattern of vectors and matrices:

**Definition 4.1** (Pattern of a Vector/Matrix)
*Let $m \in \mathbb{N}_0$, $n \in \mathbb{N}$ with $m \le n$, and let the vector $J := (j_1, \dots, j_m) \in \mathbb{N}^m$ have pairwise different entries with $1 \le j_i \le n$ for $i = 1, \dots, n$. Then we write $i \in J$ if $\exists l : j_l = i$. We call $J$ the* **pattern of a vector** *$x \in \mathbb{K}^n$, if only the components $x_{j_l}$ for $l = 1, \dots, m$, may be different from zero, i.e.*

$$x_i = 0 \; for \; i \notin J.$$

*For a vector $x = (x_1, \dots, x_n) \in \mathbb{K}^n$ that has the pattern $J = (j_1, \dots, j_m)$, we define the $J$-***reduced vector** $x(J) \in \mathbb{K}^m$ *of $x$ by*

$$x(J) := (x_{j_1}, \dots, x_{j_m})^T .$$

*We refer to $m$ as the* **length of the pattern** $J = (j_1, \dots, j_m)$ *and we write in abbreviation $\#J := m$. Further, we refer to the $l$-th entry of the pattern $J$ by $(J)_l$. If the patterns $I := (i_1, \dots, i_l)$ and $J := (j_1, \dots, j_m)$ are pairwise disjoint, we define the corresponding* **joined pattern** $\tilde{J} \in \mathbb{N}^{(m+l)}$ *by*

$$\tilde{J} := I \cup J := (i_1, \dots, i_l, j_1, \dots, j_m) .$$

*We define the relation "$J \subset I$" for two patterns $J$ and $I$ with $\#J \leq \#I$ by*

$$J \subset I :\Longleftrightarrow \quad (J)_i = (I)_i \;\; for\; i = 1, \dots, \#J,$$

*and we call $J$* **sub-pattern** *of $I$.*
*We call $S = (J_1, \dots, J_n)$* **matrix-pattern** *of the matrix $A \in \mathbb{K}^{n \times n}$, if the vectors $J_k = \left(j_1^k, \dots, j_{m_k}^k\right) \in \mathbb{N}^{m_k}$ for $k = 1, \dots, n$ are the patterns of the columns $Au_k$ of the matrix $A$, and we refer to $J_k$ as the $k$-th* **column-pattern** *of the matrix $A$.*
*If the matrix $A \in \mathbb{K}^{n \times n}$ has the pattern $S$, and if $A$ is upper triangular or diagonal, we call the matrix-pattern $S$* **upper triangular** *, or* **diagonal** *respectively.*
*Let $J_i$ be the pattern of a vector $x_i \in \mathbb{K}^n$ for $i = 1, 2$. Then, for a matrix $A \in \mathbb{K}^{n \times n}$, the $(J_1, J_2)$-***reduced matrix** $(A)(J_1, J_2) \in \mathbb{K}^{\#J_1 \times \#J_2}$ *is defined by*

$$((A)(J_1, J_2))_{ij} := (A)_{(J_1)_i, (J_2)_j} ,$$

*for $i = 1, \dots, \#J_1$, and $j = 1, \dots, \#J_2$. Let the vector $x \in \mathbb{K}^n$ have the pattern $J = (j_1, \dots, j_m)$. Then the $\#J$-dimensional subspace $\mathbb{K}^n_J$ of $\mathbb{K}^n$ defined by*

$$\mathbb{K}^n_J := \operatorname{span}(u_{j_1}, \dots, u_{j_m})$$

*is called $J$-***induced subspace** *of $\mathbb{K}^n$.*

With the above definition all technical terms for the discussion of sparse vectors and matrices with certain sparsity patterns are at hand.

We give the definition of projection methods:

**Definition 4.2** (Projection Method, [83], p. 21)
*Let $A$ be a non-singular matrix in $\mathbb{K}^{n\times n}$. A method for computing a right-hand side approximate inverse $P$ of the matrix $A$ is called* **projection method**, *if auxiliary square non-singular matrices $Z_L$, $Z_R \in \mathbb{K}^{n\times n}$ exist, such that the equations*

$$u_i^H Z_L^H \left[AP - I\right] Z_R u_k = 0 \tag{4.1}$$

*hold for $i \in J_k$ with $k = 1, \ldots, n$, where $(J_1, \ldots, J_n)$ is the matrix-pattern of the product matrix $PZ_R$. This matrix-pattern is called* **projection pattern**.
*The matrix $P$ defined by equations* (4.1) *is called* **projective approximate inverse**.
*The matrices $Z_L$ and $Z_R$ are called left-hand and right-hand side* **projection matrices**.

**The Basic Idea of Projection Methods**
Projection methods for matrices in $\mathbb{R}^{n\times n}$ were investigated by Zimmermann in [83], where projective approximate inverses with fixed sparsity patterns were tested for two particular projection methods. The projection methods considered in [83] are characterized by the choices $Z_L := Z_R := I$ and $Z_L = A$, $Z_R = I$. The latter projection method coincides with the Frobenius-norm minimizing preconditioning technique introduced by Benson in [2] (see section 6.2, and e.g. [32] and [39] for recent results).

Equation (4.1) can be considered as a weak formulation for the inverse of A: if the matrix in the brackets disappears, then $P$ equals the inverse of $A$. For the practical application as a preconditioning matrix the fill-in of the matrix $P$ will be limited, and thus the matrix $P$ will be only an approximation to $A^{-1}$.
The concept of approximating the inverse of a matrix with a projection method becomes obvious for the special case $Z_L = Z_R = I$:
Instead of solving the $(n \times n)$ linear systems

$$At_k = u_k$$

with $t_k \in \mathbb{K}^n$ and $u_k$ denoting the $k$-th unit vector in $\mathbb{K}^n$, for $k = 1, \ldots, n$, which would produce the true inverse $A^{-1} = (t_1, \ldots, t_n)$, the $(\#J_k \times \#J_k)$-linear systems

$$A\left(J_k, J_k\right) p_k\left(J_k\right) = u_k\left(J_k\right)$$

are solved for $k = 1, \ldots, n$, and the resulting matrix $P = (p_1, \ldots, p_n)$ is regarded as an approximation to $A^{-1}$ (see definition 4.1 for the definition of the $(J_k, J_k)$-reduced matrix $A\left(J_k, J_k\right)$ and the $J_k$-reduced vectors $p_k\left(J_k\right)$ and $u_k\left(J_k\right)$).

Projection methods form an entire class of methods for computing one-sided approximate inverses. The choice of the projection matrices $Z_L$ and $Z_R$ specifies the particular projection method. Despite the generality of the above definition, particular estimates on the quality of the approximation of $P$ to $A^{-1}$ are possible. We derive these estimates in section 4.2.
Note, that neither the existence, nor the uniqueness of the projective approximate inverse $P$ from definition 4.2 is guaranteed. Further, if the projective approximate inverse $P$ exists and is unique, it is unknown whether or not $P$ is non-singular. A necessary and sufficient condition for the existence and the uniqueness of the projective approximate inverse $P$ from definition 4.2 is given in lemma 4.4 and in corollary 4.5.
In the following, we consider both general projection methods with $Z_L$, $Z_R$ arbitrary and projection methods with $Z_L$ arbitrary and $Z_R$ diagonal. For projection methods with diagonal $Z_R$, the projection pattern $(J_1, \ldots, J_n)$ is already the pattern of the projective approximate inverse $P$. Importantly, projection methods with diagonal $Z_R$ are interesting for practical implementations, since they can be implemented completely in parallel.

We will use projection methods in three different ways:

1. We classify some of todays preconditioning techniques in terms of projection methods, and thus provide a new theoretical framework for these methods.

2. We derive new algorithms for computing approximate inverses by making particular choices for the projection matrices $Z_L$ and $Z_R$.

3. We propose strategies and algorithms for the adaptive determination of the sparsity pattern of approximate inverses calculated by projection methods.

**Left-Hand Side Approximate Inverses**
The formulation of projection methods given in definition 4.2 considers right-hand side approximate inverses. All considerations regarding projection methods carry over for left-hand side approximate inverses $W \in \mathbb{K}^{n\times n}$ by noting that

$$\begin{aligned} & u_i^H Z_L^H \left[WA - I\right] Z_R u_k &= 0 \\ \iff \quad & u_k^H Z_R^H \left[A^H W^H - I\right] Z_L u_i &= 0, \end{aligned}$$

i.e. the matrix $W^H$ can be regarded as a right-hand side approximate inverse of $A^H$. For simplicity, we consider right-hand side approximate inverses $P$ only.

**Explicit Representation of the Approximate Inverse**
If the projection matrices $Z_L$ and $Z_R$ are known and a projection pattern $(J_1, \ldots, J_n)$

is prescribed, under certain circumstances the projective approximate inverse $P$ can be calculated directly from (4.1). For discussing that matter, we give the following definition, which characterizes three catagories of projection methods:

**Definition 4.3** ($J_k$-Explicit, Explicit and Practical Projection Methods)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in \mathbb{K}^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in \mathbb{K}^{n\times n}$ on the projection pattern $(J_1,\dots,J_n)$ and let $k \in \mathbb{N}$ be arbitrary with $1 \leq k \leq n$.*

*i) If the $(J_k, J_k)$-reduced matrix*

$$\left(Z_L^H A\right)(J_k, J_k) \in \mathbb{K}^{\#J_k \times \#J_k} \tag{4.2}$$

*of the product matrix $Z_L^H A$ is non-singular, we call the projection method* ***$J_k$-explicit***.

*ii) If the projection method is $J_k$-explicit for all $k = 1,\dots,n$, we call the projection method* ***explicit***.

*iii) An explicit projection method is called* ***practical***, *if the $J_k$-reduced vectors $\left(Z_L^H Z_R u_k\right)(J_k) \in \mathbb{K}^{\#J_k}$ are non-vanishing for all $k = 1,\dots,n$.*

The properties of projection methods introduced in the above definition are utilized in the following lemma, which states, among other things, that the projective approximate inverses of explicit projection methods have a unique explicit representation.

**Lemma 4.4** (Explicit Representation of the Projective Approximate Inverse)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in \mathbb{K}^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in \mathbb{K}^{n\times n}$ on the projection pattern $(J_1,\dots,J_n)$.*

*i) If the projection method is $J_k$-explicit according to definition 4.3 for some $k$ with $1 \leq k \leq n$, then the $J_k$-reduced $k$-th column $(PZ_R u_k)(J_k)$ of the product matrix $PZ_R$ has the unique explicit representation*

$$(PZ_R u_k)(J_k) = \left[(Z_L^H A)(J_k, J_k)\right]^{-1} (Z_L^H Z_R u_k)(J_k), \tag{4.3}$$

*and with the matrix $Q_k$ is defined by*

$$(Q_k)_{ij} := \begin{cases} \left(\left[(Z_L^H A)(J_k, J_k)\right]^{-1}\right)_{lm} & , \text{ for } i = (J_k)_l \text{ and } j = (J_k)_m \\ 0 & , \text{ else,} \end{cases} \tag{4.4}$$

*the equality*

$$PZ_Ru_k = Q_kZ_L^HZ_Ru_k \tag{4.5}$$

*holds. In* (4.5)*, the matrix* $Q_k$ *is utilized to embed the* $(\#J_k \times \#J_k)$*-matrix*

$$\left[(Z_L^HA)(J_k, J_k)\right]^{-1}$$

*into* $I\!K^{n\times n}$ *along the pattern* $J_k$ *for its rows and columns.*
*Further, the columns* $q_i^k$ *of the matrix* $Q_k$ *from* (4.4) *with* $i \in J_k$ *form a basis of the* $J_k$*-induced subspace* $I\!K^n_{J_k}$ *of* $I\!K^n$*.*

*ii) If the projection method is explicit according to definition* 4.3*, then the projective approximate inverse* $P$ *has the unique explicit representation*

$$P = \left(Q_1Z_L^HZ_Ru_1, \ldots, Q_nZ_L^HZ_Ru_n\right) Z_R^{-1}, \tag{4.6}$$

*where the matrices* $Q_k$ *for* $k = 1, \ldots, n$ *are as defined in* (4.4)*.*

*iii) If the projection method is practical according to definition* 4.3*, then none of the columns of the product matrix* $PZ_R$ *is vanishing.*

*iv) If the projection method is explicit, but not practical, then at least one of the columns* $(PZ_R)\,u_k$ *of the product matrix* $PZ_R$ *is vanishing, and hence the projective approximate inverse* $P$ *is singular.*

**Proof.**

We denote by $(PZ_Ru_k)(J_k)$ the $J_k$-reduced $k$-th column of the matrix $PZ_R$, and by $(Z_L^HZ_Ru_k)(J_k)$ the $J_k$-reduced $k$-th column of the product matrix $Z_L^HZ_R$ for $k = 1, \ldots, n$.
We prove assertion $i$):
Let $1 \le k \le n$ be arbitrary fixed, and let the projection method be $J_k$-explicit. Then equation (4.1) corresponding to this index $k$ is equivalent to the linear system

$$(Z_L^HA)(J_k, J_k)(PZ_Ru_k)(J_k) = (Z_L^HZ_Ru_k)(J_k).$$

Since the matrix $(Z_L^HA)(J_k, J_k)$ is non-singular by assumption, we can rewrite the above equation in the form

$$(PZ_Ru_k)(J_k) = \left[(Z_L^HA)(J_k, J_k)\right]^{-1}(Z_L^HZ_Ru_k)(J_k).$$

With the matrix $Q_k$ from (4.4), we embed equations (4.3) into $I\!K^n$ and obtain

$$PZ_R u_k = Q_k Z_L^H Z_R u_k.$$

Further, since the matrix $\left(Z_L^H A\right)(J_k, J_k)$ is non-singular by assumption, the columns $q_i^k$ of the matrix $Q_k$, for $i \in J_k$ are linearly independent and form a basis of the $J_k$-induced subspace $I\!K^n_{J_k}$ of $I\!K^n$
We verify equation $ii$):
Since the projection method is explicit according to definition 4.3, equation (4.5) holds for $k = 1, \ldots, n$, i.e. we have

$$\begin{array}{rrl} & PZ_R & = (Q_1 Z_L Z_R u_1, \ldots, Q_n Z_L Z_R u_n) \\ \Longleftrightarrow & P & = (Q_1 Z_L Z_R u_1, \ldots, Q_n Z_L Z_R u_n)\, Z_R^{-1} \end{array}$$

where the matrices $Q_k$ are defined in (4.4). Since the vectors $PZ_R u_k$ are uniquely defined by assertion $i$), the above representation of the projective approximate inverse $P$ is unique.
We verify assertion $iii$):
Since the projection method is practical according to definition 4.3 by assumption, it is in particular $J_k$-explicit for all $k = 1, \ldots, n$. Since the vectors $Z_L Z_R u_k$ are non-vanishing by assumption, the right-hand sides in (4.3) are non-vanishing, and thus the vectors $PZ_R u_k$ non-vanishing as well for $k = 1, \ldots, n$.
We verify assertion $iv$):
Since the projection method is explicit, by assertion $ii$) the relation

$$PZ_R = \left(Q_1 Z_L^H Z_R u_1, \ldots, Q_n Z_L^H Z_R u_n\right)$$

holds, with the matrices $Q_k$ defined in (4.4). Since the projection method is not practical, by (4.3) at least one of the columns of the matrix $PZ_R$ is vanishing. Thus the matrix

$$\left(Q_1 Z_L^H Z_R u_1, \ldots, Q_n Z_L^H Z_R u_n\right)$$

is singular. Because the right-hand side projection matrix $Z_R$ is non-singular, we obtain that the projective approximate inverse $P$ is singular.

$\diamondsuit$

The above lemma states, that for a given projection method with the projection matrices $Z_L$ and $Z_R$ on the projection pattern $(J_1, \ldots, J_n)$ the projective approximate inverse $P$ is defined uniquely, if the $n$ small linear systems

$$\left(Z_L^H A\right)(J_k, J_k)\, t_k = \left(Z_L^H Z_R u_k\right)(J_k)$$

for $t_k \in \mathbb{K}^{\# J_k}$ and $k = 1, \ldots, n$ have unique solutions, i.e. if the considered projection method is explicit according to definition 4.3.
Further, if apart of the projection matrices $Z_L$ and $Z_R$, the inverse $Z_R^{-1}$ of $Z_R$ is known as well, the projective approximate inverse $P$ of an explicit projection method can be calculated explicitly by equation (4.6). Thus, projection methods with "simple" right-hand side projection matrices, e.g. with diagonal $Z_R$, are of particular interest for practical implementations.
For practical preconditioning with approximate inverses determined by projection methods, the projective approximate inverse $P$ should be non-singular. A necessesary condition for the non-singularity of the obtained projective approximate inverse is, that the considered projection method is not only explicit, but practical according to definition 4.3. However, the projective approximate inverse given by a practical projection method can still be singular. For particular projection methods theoretical investigations on the non-singularity of the corresponding projective approximate inverses are possible. For the projection methods considered in sections 6.3 and 6.7, statements on the non-singularity of the corresponding projective approximate inverses are given in [46].

**The `Projective Approximate Inverse` Algorithm**
Algorithm 6 gives a pseudo-code formulation for obtaining the projective approximate inverse $P$ by an explicit projection method. The input data of this algorithm are the original matrix $A$, the projection matrices $Z_L$, $Z_R$ and the projection pattern $(J_1, \ldots, J_n)$. On output, the projective approximate inverse $P$ corresponding to the chosen projection method is obtained.

1. For $k = 1, \ldots, n$
   (a) Determine the matrix
   $$\left(Z_L^H A\right)(J_k, J_k)$$
   and the vector
   $$\left(Z_L^H Z_R u_k\right)(J_k)$$
   (b) Solve the linear system
   $$\left(Z_L^H A\right)(J_k, J_k)\,(v_k)\,(J_k) = \left(Z_L^H Z_R u_k\right)(J_k)$$
2. Obtain the projective approximate inverse $P$ from
   $$P := (v_1, \ldots, v_n)\, Z_R^{-1}$$

**Algorithm 6:** `Projective Approximate Inverse`

In step (1a) of algorithm 6, the $(J_k, J_k)$-reduced matrix $\left(Z_L^H A\right)(J_k, J_k)$ and the $J_k$-reduced vector $\left(Z_L^H Z_R u_k\right)(J_k)$ are determined.s In step (1b), $n$ linear systems of the size $(\#J_k \times \#J_k)$ are solved. In step (2), the projective approximate inverse $P$ is obtained by a formal matrix-matrix product. For practical relevance, the matrix $Z_R$ must be such that its inverse $Z_R^{-1}$ is either known or easily computable.
For the practical application of the `Projective Approximate Inverse` algorithm (algorithm 6), the matrices $Z_L$ and $Z_R$ should have a particular form such that neither step (1a) nor step (2) contribute much to the numerical complexity. In this case, the computational cost is dominated by the equation solves in step (1b). Since these linear systems can be solved independently from each other, and hence in parallel, algorithm 6 is well-suited for vector and parallel computers.
The linear systems in step (1b) of algorithm 6 are non-Hermitian and not positive definite in general. The density of those linear systems depends on the particular choice of the matrix $Z_L$, on the sparsity of $A$ and on the length of the column-patterns $J_k$ for $k = 1, \ldots, n$.

**The Case of Diagonal $Z_R$**
The following corollary gives the result of lemma 4.4 for the special case that $Z_R$ is a diagonal matrix. An analogous statement for real matrices for the case $Z_R := I$ is given by Zimmerman in [83].

**Corollary 4.5** (Explicit Representation of the Projective Approximate Inverse for diagonal $Z_R$, [83], p. 23)
*Let the matrix $P$ be the projective approximate inverse on the projection pattern $(J_1, \ldots, J_n)$ of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in I\!K^{n\times n}$, where $Z_L$ is arbitrary and $Z_R$ is a diagonal matrix.*

*i) If the projection method is $J_k$-explicit, then the $k$-th column $p_k$ of the projective approximate inverse $P$ has the unique explicit representation*

$$(p_k)_j = \begin{cases} \left(\left[\left(Z_L^H A\right)(J_k, J_k)\right]^{-1} \left(Z_L^H u_k\right)(J_k)\right)_i & \text{for } j = (J_k)_i, \\ 0 & \text{else,} \end{cases} \tag{4.7}$$

*i.e. the $k$-th column $p_k$ of the projective approximate inverse $P$ has the pattern $J_k$, and the $J_k$-reduced $k$-th colum $(p_k)(J_k)$ of $P$ is the solution of the $(\#J_k \times \#J_k)$-linear system*

$$\left(Z_L^H A\right)(J_k, J_k)\,(p_k)(J_k) = \left(Z_L^H u_k\right)(J_k)\,. \tag{4.8}$$

*ii) If the projection method is explicit, then the projective approximate inverse $P$ is uniquely defined. The $k$-th column of $P$ has the representation (4.7) $k = 1, \ldots, n$.*

*iii) If the projection method is practical, then none of the columns $p_k$ of the projective approximate inverse $P$ is vanishing.*

*iv) If the projection method is explicit, but not practical, then at least one column $p_k$ of the projective approximate inverse $P$ is vanishing, and thus $P$ is singular.*

**Proof.**

We prove assertion *i*):
Let the diagonal entries of the right-hand side projection matrix $Z_R$ be denoted by $d_k$ for $k = 1, \dots, n$. We consider equation (4.3) for diagonal $Z_R$:

$$\begin{aligned}
& \left(PZ_R u_k\right)(J_k) = \left[\left(Z_L^H A\right)(J_k, J_k)\right]^{-1} \left(Z_L^H Z_R u_k\right)(J_k) \\
\Longleftrightarrow \quad & d_k \left(p_k\right)(J_k) = d_k \left[\left(Z_L^H A\right)(J_k, J_k)\right]^{-1} \left(Z_L^H u_k\right)(J_k) \\
\Longrightarrow \quad & \left(p_k\right)_j = \begin{cases} \left(\left[\left(Z_L^H A\right)(J_k, J_k)\right]^{-1} \left(Z_L^H u_k\right)(J_k)\right)_i & \text{for } j = (J_k)_i, \\ 0 & \text{else.} \end{cases}
\end{aligned}$$

Assertion *ii*) follows directly from assertion *i*), since an explicit projection method is $J_k$-explicit for all $k = 1, \dots, n$.
Assertions *iii*) and *iv*) follow directly from lemma 4.4.

$\diamondsuit$

As a result of the above corollary, in the case that $Z_R$ is a diagonal matrix, the columns $p_k$ of the projective approximate inverse $P$ can be calculated independently from each other by solving the $n$ independent small linear systems

$$(Z_L^H A)(J_k, J_k)(p_k)(J_k) = (Z_L^H u_k)(J_k),$$

for $k = 1, \dots, n$, provided that the considered projection method is explicit according to definition 4.3. Note that in this situation the columns $p_k$ of the projective approximate inverse $P$ have the patterns $J_k$ for $k = 1, \dots, n$. Thus, in this case the approximate inverse $P$ can be computed columnwise completely in parallel.
For practical preconditioning purposes, only projection methods which are practical according to definition 4.3 should be considered, since otherwise at least one column $p_k$ of the obtained projective approximate inverse $P$ is vanishing, and thus $P$ is singular.
In algorithm 7, we give a pseudo-code form of the `Projective Approximate Inverse` algorithm for diagonal $Z_R$:
In practice, the matrix $Z_L$ must have a particular form, so that the computational cost for determining the $(J_k \times J_k)$ matrices $\left(Z_L^H A\right)(J_k, J_k)$ in step (1a) of algorithm

1. For $k = 1, \ldots, n$
   (a) Determine the matrix
   $$\left(Z_L^H A\right)(J_k, J_k)$$
   and the vector
   $$\left(Z_L^H u_k\right)(J_k)$$
   (b) Solve the linear system
   $$\left(Z_L^H A\right)(J_k, J_k)\,(p_k)\,(J_k) = \left(Z_L^H u_k\right)(J_k)$$
2. Obtain the projective approximate inverse by
   $$P = (p_1, \ldots, p_n)$$

**Algorithm 7:** `Projective Approximate Inverse` with diagonal $Z_R$

7 is neglegtable compared to the computational cost for solving the linear systems in step (1b).
In each step of the $k$-loop in step (1) of algorithm 7, the column $p_k$ of the projective approximate inverse $P$ is determined by solving the linear system

$$(Z_L^H A)(J_k, J_k)(p_k)(J_k) = (Z_L^H u_k)(J_k).$$

Since the $k$-loop in step (1) of algorithm 7 can be implemented completely in parallel, the above algorithm is very appealing for the implementation on parallel computers.

## 4.2 Approximation Properties

For practical preconditioning purposes the approximate inverse utilized as the preconditioner of a linear system $Ax = b$ should be as "close" as possible to the true inverse $A^{-1}$ of the coefficient matrix A. In this section, we investigate in which respect the projective approximate inverse $P$ of a projection method approximates the true inverse $A^{-1}$ of a non-singular matrix $A$. More particularly, we derive new minimization properties for projective approximate inverses of $J_k$-explicit projection methods, which give statements on the "distance" between the projective approximate inverse $P$ and the true inverse $A^{-1}$.

First, we define some quantities which we will utilize as a measure for the distance between the projective approximate inverse and the exact inverse:

**Definition 4.6** ($J_k$-Residual and $J_k$-Error)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in I\!K^{n\times n}$ on the projection pattern $(J_1, \dots, J_n)$. Let this projection method be $J_k$-explicit according to definition 4.3 for some $k \in I\!N$ with $1 \leq k \leq n$. Then we call the quantity*

$$r_{J_k} := (AP - I)Z_R u_k \tag{4.9}$$

$J_k$**-residual** *and*

$$e_{J_k} := A^{-1} r_{J_k} = (P - A^{-1})Z_R u_k \tag{4.10}$$

$J_k$**-error** *for $k = 1, \dots, n$.*

The norms of the $J_k$-residuals and $J_k$-errors are a column-wise measure for the distance between the true and the projective approximate inverse weighted by the right-hand side projection matrix $Z_R$. If the $J_k$-residuals (or equivalently the $J_k$-errors) vanish for all $k = 1, \dots, n$, then the projective approximate inverse $P$ equals the true inverse of $A$, and thus the approximation is optimal.
In practical applications, the $J_k$-errors are not known unless the product matrix $A^{-1}Z_R$ is known as well. However, the $J_k$-residuals are known in practical applications. In chapter 5, we derive strategies for the adaptive generation of projection patterns which are based on minimizing the $J_k$-residuals defined in (4.9).

In preparation of the forthcoming approximation theorem, we have to give two technical lemmata first:
First of all, we cite a technical lemma from [80]:

**Lemma 4.7** ([80], p. 32)
*Let the matrices $M$ and $R$ be elements of $I\!R^{n\times n}$, with $M$ symmetric positive definite and $R$ skew symmetric. Let $\rho(R)$ denote the spectral radius of $R$ and $\mu_m(M)$ denote the minimal eigenvalue of $M$, then the following inequality holds:*

$$\left\|I + M^{-1}R\right\|_M^2 \leq 1 + \frac{\rho^2(R)}{\mu_m^2(M)}. \tag{4.11}$$

The following lemma provides us with some insight on the properties of $J_k$-residuals:

**Lemma 4.8**
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in I\!K^{n\times n}$ on the projection pattern $(J_1, \dots, J_n)$. Let this projection method be $J_k$-explicit for some $k \in I\!N$ with $1 \le k \le n$. Then for the $J_k$-residuals of this projection method the equalities*

$$r_{J_k}^H Z_L w = 0 \tag{4.12}$$

*and*

$$r_{J_k}^H Z_L A^{-1} r_{J_k} = r_{J_k}^H Z_L A^{-1} \left(Aw - Z_R u_k\right) \tag{4.13}$$

*hold for all vectors $w \in I\!K_{J_k}^n$, the $J_k$-induced subspace of $I\!K^n$ according to definition 4.1.*

**Proof.**

We verify assertion (4.12):
The $\#J_k$ unit vectors $u_i \in I\!K^n$ with $i \in J_k$ form a basis of the $J_k$-induced subspace $I\!K_{J_k}^n$. For $i \in J_k$ we have

$$r_{J_k}^H Z_L u_i \overset{(4.9)}{=} \left(u_i^H Z_L^H (AP - I) Z_R u_k\right)^H \overset{(4.1)}{=} 0. \tag{4.14}$$

Thus we have, with $\alpha_i \in I\!K$ for $i \in J_k$ for arbitrary $w := \sum_{i \in J_k} \alpha_i u_i \in I\!K_{J_k}^n$:

$$r_{J_k}^H Z_L w = \sum_{i \in J_k} \alpha_i r_{J_k}^H Z_L u_i \overset{(4.14)}{=} 0,$$

and the proof of assertion (4.12) is complete.

We prove assertion (4.13):

We apply lemma 4.4 to rewrite $r_{J_k}$:

$$\begin{aligned} r_{J_k} &\overset{(4.9)}{=} (AP - I) Z_R u_k \\ &\overset{(4.5)}{=} A Q_k Z_L^H Z_R u_k - Z_R u_k, \end{aligned} \tag{4.15}$$

with the matrices $Q_k$ defined in (4.4).

We denote for $i = 1, \ldots, \#J_k$ by $q_i^k$ the $(J_k)_i$-th column of the matrix $Q_k$. Note that, since the considered projection method is $J_k$-explicit, assertion $i)$ of lemma 4.4 states that the vectors $\left(q_1^k, \ldots, q_{\#J_k}^k\right)$ form a basis of the $J_k$-induced subspace $\mathbb{K}_{J_k}^n$, whereas all other columns of $Q_k$ are vanishing. Thus, we can rewrite equation (4.15):

$$r_{J_k} = \sum_{i \in J_k} (Z_L^H Z_R u_k)_i A q_i^k - Z_R u_k, \tag{4.16}$$

where $\left(Z_L^H Z_R u_k\right)_i$ denotes the $i$-th component of the vector $Z_L^H Z_R u_k$, and we obtain

$$\begin{aligned}
r_{J_k}^H Z_L A^{-1} r_{J_k} &= r_{J_k}^H Z_L A^{-1} \left( \sum_{i \in J_k} (Z_L^H Z_R u_k)_i A q_i^k - Z_R u_k \right) \\
&= r_{J_k}^H Z_L \left( \sum_{i \in J_k} \underbrace{(Z_L^H Z_R u_k)_i q_i^k}_{\in \mathbb{K}_{J_k}^n} - A^{-1} Z_R u_k \right) \\
&\overset{(4.12)}{=} r_{J_k}^H Z_L \left( \sum_{i \in J_k} \eta_i q_i^k - A^{-1} Z_R u_k \right),
\end{aligned}$$

where $\eta_i$ are arbitrary elements of $\mathbb{K}$. Since the vectors $q_i^k$ with $i \in J_k$ form a basis of $\mathbb{K}_{J_k}^n$, assertion (4.13) is proved.

$\diamondsuit$

The following theorem gives a measure for the size of the $J_k$-residuals and $J_k$-errors for an approximate inverse $P$ determined by a projection method.

**Theorem 4.9** (Approximation Theorem for Projection Methods)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in \mathbb{K}^{n \times n}$ determined by a projection method with the projection matrices $Z_L$, $Z_R \in \mathbb{K}^{n \times n}$ on the projection pattern $(J_1, \ldots, J_n)$.*

*i) Let $\mathbb{K}^n = \mathbb{R}^n$ and let the projection method be $J_k$-explicit for some $k \in \mathbb{N}$ with $1 \leq k \leq n$. Let the matrix $Z_L A^{-1} \in \mathbb{R}^{n \times n}$ be positive real. Let $Z_L A^{-1} = M + R$, where $M$ is the symmetric part and $R$ is the skew symmetric part of $Z_L A^{-1}$. Let $\rho(R)$ denote the spectral radius of $R$, and let $\mu_m(M)$ denote the minimal eigenvalue of $M$. Let $P$ be given by equation (4.1). Then the following estimations hold:*

$$\left\| (AP - I) Z_R u_k \right\|_{Z_L A^{-1}} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \cdot \min_{w \in \mathbb{R}_{J_k}^n} \left\| Aw - Z_R u_k \right\|_{Z_L A^{-1}}, \tag{4.17}$$

$$\left\|(P - A^{-1})Z_R u_k\right\|_{A^T Z_L} \le \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \cdot \min_{w \in I\!R^n_{J_k}} \left\|w - A^{-1} Z_R u_k\right\|_{A^T Z_L} . \tag{4.18}$$

*ii) Let $I\!K^n \in \{I\!R^n, \mathbb{C}^n\}$ and let the matrix $Z_L A^{-1} \in I\!K^{n \times n}$ be Hermitian positive definite. Then the equalities*

$$\left\|(AP - I)Z_R u_k\right\|_{Z_L A^{-1}} = \min_{w \in I\!K^n_{J_k}} \left\|Aw - Z_R u_k\right\|_{Z_L A^{-1}} , \tag{4.19}$$

$$\left\|(P - A^{-1})Z_R u_k\right\|_{A^H Z_L} = \min_{w \in I\!K^n_{J_k}} \left\|w - A^{-1} Z_R u_k\right\|_{A^H Z_L} \tag{4.20}$$

*hold for all $k = 1, \ldots n$.*

**Proof.**

First, we verify inequality (4.17):
By lemma 4.8, we can write

$$\left\|r_{J_k}\right\|^2_{Z_L A^{-1}} = r^T_{J_k} Z_L A^{-1} \left(Aw - Z_R u_k\right), \tag{4.21}$$

where the vector $w \in I\!R^n_{J_k}$ is arbitrary. We choose the vector $w$ such that the vector $v_k \in I\!R^n$ defined by $v_k = Aw - Z_R u_k$ satisfies

$$\left\|v_k\right\|_{Z_L A^{-1}} = \min_{w \in I\!R^n_{J_k}} \left\|Aw - Z_R u_k\right\|_{Z_L A^{-1}} . \tag{4.22}$$

With $Z_L A^{-1} = M + R = M(I + M^{-1}R)$ and lemma 4.7, we have

$$\begin{aligned}
\left\|r_{J_k}\right\|^2_{Z_L A^{-1}} &= r^T_{J_k} M(I + M^{-1}R) v_k \\
&\le \left\|r_{J_k}\right\|_M \left\|(I + M^{-1}R) v_k\right\|_M \\
&\quad \text{by the Cauchy-Schwarz inequality} \\
&\le \left\|r_{J_k}\right\|_M \left\|I + M^{-1}R\right\|_M \left\|v_k\right\|_M \\
&\overset{(4.11)}{\le} \left\|r_{J_k}\right\|_M \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \left\|v_k\right\|_M .
\end{aligned}$$

Division by $\left\|r_{J_k}\right\|_{Z_L A^{-1}}$ completes the proof for inequality (4.17).

The corresponding result for inequality (4.18) follows trivially with $r_{J_k} = Ae_{J_k}$ and

$$\left\|r_{J_k}\right\|^2_{Z_L A^{-1}} = \left\|Ae_{J_k}\right\|^2_{Z_L A^{-1}} = e^T_{J_k} A^T Z_L A^{-1} A e_{J_k} = \left\|e_{J_k}\right\|^2_{A^T Z_L} . \tag{4.23}$$

We prove equality (4.19):
Note that since the matrix $Z_L A^{-1}$ is Hermitian positive definite, the matrix $Z_L^H A$ is Hermitian positive as well. Hence, all corresponding $(J_k, J_k)$-reduced matrices $\left(Z_L^H A\right)(J_k, J_k)$ are non-singular, and consequently the considered projection method is explicit.
In the following, we assume $k \in I\!N$ with $1 \leq k \leq n$ arbitrary fixed. With lemma 4.8, we have

$$\left\|r_{J_k}\right\|^2_{Z_L A^{-1}} = r^H_{J_k} Z_L A^{-1} \left(Aw - Z_R u_k\right)$$

with the arbitrary vector $w \in I\!K^n_{J_k}$. We choose $w$ such that the vector $v_k \in I\!K^n$ defined by $v_k = Aw - Z_R u_k$ satisfies

$$\left\|v_k\right\|_{Z_L A^{-1}} = \min_{w \in I\!K^n_{J_k}} \left\|Aw - Z_R u_k\right\|_{Z_L A^{-1}} . \tag{4.24}$$

Analogous to the real case, we have

$$\begin{aligned} \left\|r_{J_k}\right\|^2_{Z_L A^{-1}} &= r^H_{J_k} Z_L A^{-1} v_k \\ &\leq \left\|r_{J_k}\right\|_{Z_L A^{-1}} \left\|v_k\right\|_{Z_L A^{-1}} , \end{aligned}$$

by the Cauchy-Schwarz inequality. By division with $\left\|r_{J_k}\right\|_{Z_L A^{-1}}$ and by noting that the $J_k$-residual $r_{J_k}$ has by (4.9) the form $APZ_R u_k - Z_R u_k$ where $PZ_R u_k \in I\!K^n_{J_k}$, with (4.24) the proof for equality (4.19) is complete.
Assertion (4.20) follows analogously to (4.23).

$\diamondsuit$

The theorem above is an adaption of theorem 3.7 derived by Weiss ([80], pp. 32 – 33) in the context of orthogonalization methods.
Theorem 4.9 links the orthogonality conditions of equations (4.1) to a minimization property for the $J_k$-residuals and $J_k$-errors. In this sense, the above theorem establishes a statement on the quality of the approximation of the projective approximate inverse $P$ determined by a $J_k$-explicit projection method to the true inverse $A^{-1}$. More precisely: for Hermitian positive definite $Z_L A^{-1}$, the vectors $PZ_R u_k \in I\!K^n_{J_k}$ minimize the $Z_L^H A^{-1}$-norm of the quantities $Aw - u_k$, where $w \in I\!K^n_{J_k}$, among the $(\#J_k)$-dimensional $J_k$-induced subspaces $I\!K^n_{J_k}$ for $k = 1, \ldots, n$.

**The Case of Diagonal $Z_R$**

For $J_k$-explicit projection methods the case that $Z_R$ is a diagonal matrix is of special interest, since in this case the columns $p_k$ of the projective approximate inverse $P$ can be computed completely in parallel. In this case, the $J_k$-residuals from definition 4.6 have the form $r_{J_k} = d_k\,(Ap_k - u_k)$, where $d_k$ denotes the $k$-th diagonal element of $Z_R$, and the columns $p_k$ of the projective approximate inverse have the pattern $J_k$ for $k = 1, \ldots, n$.

We give the approximation properties of theorem 4.9 for the special case of diagonal $Z_R$:

**Corollary 4.10** (Approximation Theorem for Projection Methods with diagonal $Z_R$)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in \mathbb{K}^{n\times n}$ determined by a projection method with the projection matrices $Z_L \in \mathbb{K}^{n\times n}$ arbitrary and $Z_R$ diagonal, on the projection pattern $(J_1, \ldots, J_n)$.*

*i) Let $\mathbb{K}^n = \mathbb{R}^n$ and let the projection method be $J_k$-explicit for some $k$ with $1 \le k \le n$. Let the matrix $Z_L A^{-1} \in \mathbb{R}^{n\times n}$ be positive real and let $Z_L A^{-1} = M + R$, where $M$ is the symmetric and $R$ is the skew-symmetric part of $Z_L A^{-1}$. Let $\rho(R)$ denote the spectral radius of $R$ and let $\mu_m(M)$ denote the minimal eigenvalue of $M$. Then the estimations*

$$\left\|Ap_k - u_k\right\|_{Z_L A^{-1}} \le \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}}\ \min_{w\in\mathbb{R}^n_{J_k}} \left\|Aw - u_k\right\|_{Z_L A^{-1}}, \tag{4.25}$$

$$\left\|p_k - A^{-1}u_k\right\|_{A^T Z_L} \le \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}}\ \min_{w\in\mathbb{R}^n_{J_k}} \left\|w - A^{-1}u_k\right\|_{A^T Z_L} \tag{4.26}$$

*hold, where $p_k$ denotes the $k$-th column of the projective approximate inverse $P$.*

*ii) Let $\mathbb{K}^n \in \{\mathbb{R}^n, \mathbb{C}^n\}$. If $Z_L A^{-1} \in \mathbb{K}^{n\times n}$ is Hermitian positive definite, the equalities*

$$\left\|Ap_k - u_k\right\|_{Z_L A^{-1}} = \min_{w\in\mathbb{K}^n_{J_k}} \left\|Aw - u_k\right\|_{Z_L A^{-1}}, \tag{4.27}$$

$$\left\|p_k - A^{-1}u_k\right\|_{A^H Z_L} = \min_{w\in\mathbb{K}^n_{J_k}} \left\|w - A^{-1}u_k\right\|_{A^H Z_L} \tag{4.28}$$

*hold for all $k = 1, \ldots, n$, where $p_k$ denotes the $k$-th column of the projective approximate inverse $P$.*

**Proof.**

From theorem 4.9 with diagonal $Z_R$ we obtain

$$\left\|(AP-I)\,Z_R u_k\right\|_{Z_L A^{-1}} \leq \sqrt{1+\frac{\rho^2(R)}{\mu_m^2(M)}}\min_{v\in \mathbb{R}^n_{J_k}} \left\|Av - Z_R u_k\right\|_{Z_L A^{-1}}$$

for positive real $Z_L A^{-1}$, and

$$\left\|(AP-I)\,Z_R u_k\right\|_{Z_L A^{-1}} = \min_{v\in \mathbb{K}^n_{J_k}} \left\|Av - Z_R u_k\right\|_{Z_L A^{-1}},$$

for Hermitian positive definite $Z_L A^{-1}$.
By noting that

$$\left\|(AP-I)\,Z_R u_k\right\|_{Z_L A^{-1}} = \left\|d_k\,(Ap_k - u_k)\right\|_{Z_L A^{-1}} = |d_k|\left\|Ap_k - u_k\right\|_{Z_L A^{-1}}$$

and

$$\begin{aligned} &\min_{v\in \mathbb{K}^n_{J_k}} \left\|Av - Z_R u_k\right\|_{Z_L A^{-1}} \\ &= \min_{w\in \mathbb{K}^n_{J_k}} \left\|d_k\,(Aw - u_k)\right\|_{Z_L A^{-1}} \\ &= |d_k| \min_{w\in \mathbb{K}^n_{J_k}} \left\|Aw - u_k\right\|_{Z_L A^{-1}} \end{aligned}$$

where $d_k$ denotes the $k$-th diagonal entry in $Z_R$, the proof for assertions (4.25) and (4.27) is complete.
Assertions (4.26) and (4.28) follow trivially by noting that

$$\left\|Ax\right\|^2_{Z_L A^{-1}} = \left\|x\right\|^2_{A^H Z_L},$$

for $x \in \mathbb{K}^n$.

$\diamondsuit$

The above corollary establishes a statement on the quality of the approximation of the $k$-th column $p_k$ of the projective approximate inverse $P$ to the $k$-th column of the true inverse $A^{-1}$. For Hermitian positive definite $Z_L A^{-1}$, the $k$-th column $p_k$ of $P$ is the minimizer of the $Z_L^H A^{-1}$-norm of the quantities $Aw - u_k$ among all vectors $w \in \mathbb{K}^n_{J_k}$, i. e. among all vectors $w$ in $\mathbb{K}^n$ that have the pattern $J_k$ for $k = 1, \ldots, n$.

**Two Examples for Projection Methods**
We demonstrate the potential of the theoretical framework of projection methods by considering the two particular projection methods explored in [83].
Let the matrix $P \in I\!K^{n\times n}$ be the projective approximate inverse of the matrix $A$ from (3.1) determined by a projection method with the projection matrices $Z_L := A$, or $Z_L := I$, and $Z_R := I$ on the projection pattern $(J_1, \ldots, J_n)$. Then the following approximation estimates are obtained:

$\underline{Z_L = A,\ Z_R = I}$**:** As shown in [83], this projection method coincides with the Frobenius-norm approach for calculating approximate inverses introduced by Benson in [2] (see section 6.2 for a more detailed discourse). The Frobenius-norm approach for calculating approximate inverses is based on minimizing the Frobenius-norm expression

$$\left\|AP - I\right\|_F^2 = \sum_{k=1}^{n} \left\|Ap_k - u_k\right\|_2^2 ,$$

where $p_k$ denotes the $k$-th column of the approximate inverse $P$, the approximate inverse $P$ has the prescribed pattern $(J_1, \ldots, J_n)$, and $u_k$ denotes the $k$-th unit-vector in $I\!K^n$. The columns $p_k$ of $P$ are the solutions of the $n$ independent $(J_k \times J_k)$ least-squares problems

$$\min_{w \in I\!K^n_{J_k}} \left\|Aw - u_k\right\|_2^2$$

for $k = 1, \ldots, n$.
Note that this projection method is always explicit according to definition 4.3, since the product matrix $Z_L^H A = A^H A$ is Hermitian positive definite.
The identities

$$\begin{aligned} \left\|Ap_k - u_k\right\|_2 &= \min_{w \in I\!K^n_{J_k}} \left\|Aw - u_k\right\|_2 , \\ \left\|p_k - A^{-1}u_k\right\|_{A^H A} &= \min_{w \in I\!K^n_{J_k}} \left\|w - A^{-1}u_k\right\|_{A^H A} \end{aligned}$$

for $k = 1, \ldots, n$, obtained from corollary 4.10 for this approach, coincide with the construction principle for this preconditioning technique.
This projection method is practical according to definition 4.3 whenever the $J_k$-reduced vectors $\left(A^H\right)(J_k)$ are non-vanishing for all $k = 1, \ldots, n$.

$\underline{Z_L = I,\ Z_R = I}$**:** This projection method is considered in [46] for fixed projection patterns. If the matrix $A^{-1}$ is positive real and if this projection method is

$J_k$-explicit according to definition 4.3, the estimates

$$\left\|Ap_k - u_k\right\|_{A^{-1}} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in I\!R^n_{J_k}} \left\|Aw - u_k\right\|_{A^{-1}},$$

$$\left\|p_k - A^{-1}u_k\right\|_{A^T} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in I\!R^n_{J_k}} \left\|w - A^{-1}u_k\right\|_{A^T}$$

hold, where $p_k$ denotes the $k$-th column of the projective approximate inverse $P$, $M$ denotes the symmetric part and $R$ denotes the skew-symmetric part of $A^{-1}$.

If the matrix $A$ is Hermitian positive definite, this projection method is explicit according to definition 4.3, and the identities

$$\left\|Ap_k - u_k\right\|_{A^{-1}} = \min_{w \in I\!K^n_{J_k}} \left\|Aw - u_k\right\|_{A^{-1}},$$

$$\left\|p_k - A^{-1}u_k\right\|_{A^H} = \min_{w \in I\!K^n_{J_k}} \left\|w - A^{-1}u_k\right\|_{A^H}$$

hold for $k = 1, \ldots, n$. Thus, for this projection method the $J_k$-residuals $Ap_k - u_k$ are minimized in the energy norm $\left\|.\right\|_{A^{-1}}$, and the $J_k$-errors are minimized in the vector norm $\left\|.\right\|_{A^H}$.

Note that if the matrix $A$ is Hermitian positive definite, this projection method is practical according to definition 4.3 whenever the relation $k \in J_k$ holds for all $k = 1, \ldots, n$. If the matrix $A$ is positive real, the non-singularity of the $(J_k, J_k)$-reduced matrices

$$(A)\,(J_k, J_k)\,,$$

and hence the $J_k$-explicitness of this projection method depends on the particular choice of the pattern $(J_1, \ldots, J_n)$.

Importantly, based on the new minimization statements given above, for this projection method new adaptive pattern derivation strategies can be developed. This issue is discussed in chapter 5 for general projection methods and in section 6.3 for this particular projection method.

In this section, we have scrutinized the general concept of calculating approximate inverses by projection methods on fixed projection patterns and we have introduced critetria and algorithms for the explicit calculation of projective approximate inverses. Furthermore, we have developed new statements on the approximation of projective approximate inverses to the corresponding exact inverses.

In practical applications, CPU-time and memory restrictions limit the number of non-zeros in the approximate inverse. Thus, the shape of the projection pattern is important for the efficiency of a projective approximate inverse applied as a preconditioner for the iterative solution of a linear system. In the following chapter, we develop strategies for the adaptive generation of such projection patterns. These strategies are essentially based on the new approximation statements given in theorem 4.9.

# 5 Adaptive Pattern Derivation for Projection Methods

In this chapter, we develop new strategies for the adaptive pattern derivation for general projection methods.

For the practical application of a projective approximate inverse as preconditioner of a linear system, sparsity is an important task. On the one hand, memory and CPU-time limitations restrict the amount of fill-in in the projective approximate inverse. On the other hand, a denser projective approximate inverse may accelerate the convergence of the iterative solve more effectively. Since in general no information of a "good" matrix-pattern for the projective approximate inverse is available, it must be determined adaptively.

Adaptive pattern derivation strategies for the Frobenius norm minimizing preconditioning technique (see section 6.2) has been introduced by Cosgrove, Diaz and Griewank in [17]. This preconditioning technique has been classified in terms of projection methods by Zimmermann in [83].

By extending the idea of Cosgrove, Diaz and Griewank to the general case, i.e. to general projection methods, we develop a new framework for the adaptive pattern derivation for projection methods. We state the new pattern adaption strategies for projection methods in form of pseudo-code algorithms. Furthermore, we discuss control strategies and the computational complexity of these new algorithms.

## 5.1 The Basic Concept of the Pattern Derivation

We consider the linear system $Ax = b$, with $A \in I\!\!K^{n\times n}$ non-singular and $x, b \in I\!\!K^n$. Suppose that the projective approximate inverse $P^0$ of the matrix $A$ is computed on some projection pattern $(J_1^0, \ldots, J_n^0)$ by an explicit projection method. The general design of our pattern derivation methods consists in generating a sequence of augmented column-patterns

$$J_k^0 \subset J_k^1 \subset \cdots \subset J_k^l, \tag{5.1}$$

for $k = 1, \ldots, n$, and the corresponding projective approximate inverses $P^1, \ldots, P^l$ of the matrix $A$, until some stopping criterion is satisfied. This strategy rests upon the following observation based on theorem 4.9:

The $Z_L A^{-1}$-norm of the $J_k^i$-residuals $r_{J_k^i}$ decreases monotonously for growing column-patterns $J_k^i$. If all column-patterns are full, i.e. $J_k^l = (1, \ldots, n)$ for $k = 1, \ldots, n$, then $P^l$ equals the true inverse $A^{-1}$, and all corresponding $J_k^l$-residuals vanish. In this sense, the approximate inverse $P^i$ with $i \geq 1$ computed with an explicit projection method on the augmented projection pattern $(J_1^i, \ldots, J_n^i)$ is a better approximation to the true inverse – and hence likely to be a more effective preconditioner – than the preceeding approximate inverses $P^0, \ldots, P^{i-1}$.

With the above considerations, projection methods can be regarded as iterative methods for approximating the inverse $A^{-1}$ of the matrix $A$: each iteration step consists of augmenting the column-patterns, and calculating the new iterate, which is the projective approximate inverse $P^l$ corresponding to the new projection pattern. In each iteration step $l$ the corresponding $J_k^l$-residuals $r_{J_k}^l$ are minimized as stated by theorem 4.9. This iteration process is finite, i.e. it terminates after at most $n$ steps: if all column-patterns are full, the corresponding projective approximate inverse $P$ equals the true inverse $A^{-1}$.

Note that the projection patterns $\left(J_0^l, \ldots, J_n^l\right)$, with $0 \leq l \leq n$, from relation (5.1) must be choosen in such a way that the corresponding projection methods are explicit. Otherwise, the corresponding projective approximate inverse cannot be calculated explicitly with lemma 4.4.

For expounding the general design of our pattern derivation strategy in greater detail, we consider the situation that the augmented pattern $J_k^i$ is obtained by adding one new index $j$ to the known pattern $J_k^{i-1}$, i.e. $J_k^i := J_k^{i-1} \cup (j)$. Theoretically, for enlarging the column-pattern $J_k^{i-1}$ any index $j$ with $1 \leq j \leq n$ and $j \notin J_k^{i-1}$ can be chosen, provided that the projection method is $J_k^i$-explicit. Since the approximate inverse must be sparse for practical implementations, the length of the column-patterns is restricted. Thus, the index $j$ added to the column-pattern $J_k^{i-1}$ should be chosen such that the $Z_L A^{-1}$-norm of the new $J_k^i$-residual $r_{J_k^i}$ decreases as much as possible.
Theoretically, for this purpose the $J_k^{i-1} \cup (j)$-residuals for all candidate indices $j \notin J_k^{i-1}$ could be computed directly. The augmented column-pattern could then be chosen according to the smallest $Z_L A^{-1}$-norm of the new residual $r_{J_k^{i-1} \cup (j)}$. But this strategy involves a prohibitively high computational cost. A more efficient strategy is to determine the decrease of the $J_k^{i-1}$-residual norm caused by adding an index $j \notin J_k^{i-1}$.
The following definition introduces a quantity that we will utilize for the derivation of an adaptive pattern enlarging criterion.

**Definition 5.1** (Candidate Set, Decrease Rate)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in \mathbb{K}^{n \times n}$ determined by a projection method with the projection matrices $Z_L$, $Z_R \in \mathbb{K}^{n \times n}$ on the projection pattern $(J_1, \ldots, J_n)$. Let this projection method be $J_k$-explicit for some $1 \leq k \leq n$ according to definition 4.3. For $j \in \mathbb{N}$ with $1 \leq j \leq n$ we define the* **candidate set** *$C_{J_k}$ of the column-pattern $J_k$ by*

$$C_{J_k} := \left\{ j \mid j \notin J_k,\ \left(Z_L^H A\right) (J_k \cup (j), J_k \cup (j)) \textit{ is non-singular} \right\}, \tag{5.2}$$

*i.e. the candidate set $C_{J_k}$ contains all indices $j$ that may be added to the current column pattern $J_k$. For all $j \in C_{J_k}$ we define the non-negative real numbers $\lambda_j$ by*

$$\lambda_j := \left\|r_{J_k}\right\|^2_{Z_L A^{-1}} - \left\|r_{J_k \cup (j)}\right\|^2_{Z_L A^{-1}} . \tag{5.3}$$

*The $\lambda_j$ are called* **decrease rates** *of the $J_k$-residual $r_{J_k}$ for the index $j$.*

The decrease rates $\lambda_j$ from the above definition are a measure for the decrease of the $J_k$-residual norm caused by adding one candidate index $j$ to the current column-pattern $J_k$. Thus, with the decrease rates $\lambda_j$ from the above definition, a criterion for enlarging a given column pattern with regards to the optimal decrease of the $J_k$-residuals is at hand:
After computing the decrease rates $\lambda_j$ for all candidate indices $j \in C_{J_k^{i-1}}$, the enlarged column pattern $J_k^i$ is defined as $J_k^i := J_k^{i-1} \cup (j_0)$, where the index $j_0$ corresponds to the largest decrease rate $\lambda_{j_0}$, i.e. the new index $j_0$ is chosen such that

$$\lambda_{j_0} = \max_{j \notin J_k^{i-1}} \lambda_j$$

holds.
Instead of computing the exact decrease-rates $\lambda_j$, estimates $\theta_j$ for the $\lambda_j$ can be used for the pattern-derivation. This may lead to less decrease of the residual norms, but may be computationally cheaper. We introduce a strategy for computing estimates $\theta_j$ of the decrease rates $\lambda_j$ in section 5.3.

We postpone all considerations regarding the actual computation of the decrease-rates $\lambda_j$, and the corresponding estimates $\theta_j$, to sections 5.2 and 5.3, and focus on the general design of the pattern derivation algorithm:
Once the enlarged column patterns $J_k^i$, with $k = 1, \ldots, n$ are determined, a new approximate inverse $P^i$ of the matrix $A$ can be computed on the enlarged projection pattern. After that, the corresponding $J_k^i$-residuals $r_{J_k^i}$ can be computed, and the column-patterns can be enlarged once more.

**Stopping Criteria for the Adaptive Pattern Derivation**
Since the length of the column-patterns $J_k^i$ must be limited in practical applications, a stopping criterion for enlarging the column-patterns is needed. Various stopping criteria are possible:

**quantitative:** The most obvious stopping criterion consists in restricting the maximum length of the column-patterns. With this strategy both, the CPU-time and the memory required for the pattern derivation process, can be estimated in advance. On the other hand, the adaptivity of the pattern derivation process is diminished. In practical applications, the true inverse can have columns of widely differing density. Thus, some column-patterns $J_k^i$ might be larger than necessary, while others might be too small. In the first case, CPU-time is wasted, and in the second case, the convergence of the preconditioned linear system might suffer.

**qualitative:** A more adaptive stopping criterion is obtained by controlling the quality of the approximate inverse with regards to its approximation to the true inverse by monitoring the $Z_L A^{-1}$-norms of the $J_k^i$-residuals. The derivation process of the column-patterns $J_k^i$ is stopped, when the corresponding $J_k^i$-residuals become smaller than a prescribed threshold. The advantage of this strategy is, that the work devoted to the particular column-patterns $J_k^i$ depends on the structure of the true inverse. A drawback of this strategy is, that the CPU-time and memory required for the pattern derivation process cannot be estimated in advance. This problem may be alleviated by limiting the maximum number of pattern derivation steps in advance.

A further problem of the qualitative stopping criterion is that it depends on the particular choice of the projection matrices $Z_L$ and $Z_R$, whether or not the $Z_L A^{-1}$-norm of the $J_k^i$-residual $r_{J_k^i}$ can be computed at all. For example, with $Z_L = Z_R = I$, the $Z_L A^{-1}$-norm of the $J_k^i$-residual cannot be computed unless the matrix $A^{-1}$ is known. Thus, the quantity $\left\|r_{J_k^i}\right\|_{Z_L A^{-1}}$ possibly cannot be applied directly as a stopping criterion. This problem can be circumvented by simply using another norm, for instance the Euclidian norm, for measuring the $J_k^i$-residuals. But then, the norms of the $J_k^i$-residuals do not necessarily decrease monotonously as the column-patterns $J_k^i$ are enlarged, making this stopping criterion somewhat unreliable. Thus, instead of monitoring the original $J_k^i$-residuals in any norm, we suggest to monitor the related Residual-Minimizing smoothed $J_k^i$-residuals $s_{J_k^i}$ instead (see section 3.4). The smoothed $J_k^i$-residuals decrease monotonously in the Euclidian norm for growing column patterns $J_k^i$.

**combined:** A third stopping criterion is obtained by combining the quantitative and the qualitative stopping criterion: the pattern derivation is stopped when either the maximum fill-in is reached or the norm of the $J_k^i$-residual becomes smaller than a prescribed threshold value. This strategy combines the advantages of both the quantitative and the qualitative control.

**The `Pattern Adaptive Projective Approximate Inverse` Algorithm**

In order to give a pseudo-code of the pattern derivation algorithm in a general form, we define some control parameters first:

i) $\underline{mf}$ : The maximum number of indices in the column-pattern $J_k^i$. The larger this number is, the more accurate the approximate inverse will be. In order to restrict the CPU-time and memory for the construction of the approximate inverse, this number should be small in some sense.

ii) $\underline{\epsilon_k}$ : Threshold for the norm of the $J_k^i$-residual $r_{J_k^i}$ (or the Residual-Minimizing smoothed $J_k^i$-residual $s_{J_k^i}$). If

$$\left\| r_{J_k^i} \right\|_{Z_L A^{-1}} < \epsilon_k \text{ (or } \left\| s_{J_k^i} \right\|_2 < \epsilon_k\text{)},$$

the pattern adaption process for this column is stopped. The smaller this threshold is, the higher the computational cost will be, because more fill-in for the reduction of the $J_k^i$-residuals is required.

iii) $\underline{ms}$ : The maximum number of updating steps for each column-pattern. The more updating steps are performed, the higher both the accuracy of the projective approximate inverse and the computational cost will be.

iv) $\underline{mfps}$ : The maximum number of indices added in one updating step to the column-pattern. If more than one new index is added to the pattern in each step, the computational cost can be reduced, but the obtained approximate inverse might become more inaccurate – in the sense of theorem 4.9 –, too. Note that if more than one index is added to the current pattern, the non-singularity of the corresponding reduced matrix on the augmented pattern from definition 4.3 is not guaranteed by theory, and thus the corresponding projective approximate inverse $P$ may be not well-defined. However, our numerical experiments indicate, that this scruples are not necessary in practical applications. Although we allowed more than one new index per step in many cases, we never encountered a singular reduced matrix.

The further input data of the `Pattern Adaptive Projective Approximate Inverse` algorithm are the non-singular matrix $A$ for which the projective approximate inverse $P$ is sought, the projection matrices $Z_L$ and $Z_R$ from definition 4.2 and an initial projection pattern $(J_1^0, \ldots, J_n^0)$. Note that the initial pattern must be chosen in such a way that the considered projection method is explicit according to definition 4.3. For practical applications, it is imperative that the initial pattern is chosen in such a way that the projection method is practical according to definition 4.3, because otherwise the obtained approximate inverse $P$ is singular.
The output of the `Pattern Adaptive Projective Approximate Inverse` algorithm is a projective approximate inverse computed by a projection method on an adaptively determined projection pattern. Note that if the considered projection method is practical for the initial pattern, it is practical for the adaptively determined pattern as well.

1. Compute the projective approximate inverse $P^0$ with the initial projection pattern $(J_1^0, \ldots, J_n^0)$ using algorithm 6
2. $r_{J_k^0} := (AP^0 - I)\, Z_R u_k$ for $k = 1, \ldots, n$
3. For $l = 1, \ldots, ms$
   (a) For $k = 1, \ldots, n$
      i. If $\left(\left\|r_{J_k^{l-1}}\right\|_{Z_L A^{-1}} < \epsilon_k\right)$ Cycle $k$-loop
      ii. $mt := \min\left(mfps, mf - \left(\#J_k^{l-1}\right)\right)$
      iii. If $(mt == 0)$ Cycle $k$-loop
      iv. Compute the decrease rates $\lambda_j$ for all elements $j$ of the candidate set $C_{J_k^{l-1}}$ (see definition (5.1))
      v. Determine the $mt$ indices $j_1, \ldots, j_{mt}$ according to the largest decrease rates $\lambda_j$
      vi. $J_k^l := J_k^{l-1} \cup (j_1, \ldots, j_{mt})$
   (b) If no column-pattern was enlarged in step (3a) : STOP
   (c) Compute the approximate inverse $P^l$ with the projection pattern $\left(J_1^l, \ldots, J_n^l\right)$ using algorithm 6
   (d) If $(l \neq ms)$: Compute the new $J_k^l$-residuals $r_{J_k^l}$ for $k = 1, \ldots, n$

**Algorithm 8:** `Pattern Adaptive Projective Approximate Inverse`

In step (1) of algorithm 8, the initial projective approximate inverse $P^0$ is determined on the initial projection pattern. In step (2), the $J_k^0$-residuals $r_{J_k^0}$ are calculated. The actual pattern adaption process begins with step (3). In step (3(a)i), the $J_k^l$-residual norms are computed for the qualitative control of the size of the column-patterns $J_k^l$ (alternatively, the Euclidian norms of the corresponding Residual-Minimizing smoothed $J_k^l$-residuals $s_{J_k^l}$ can be utilized, see the discussion on pages 67–68). In each step of the $l$-loop, the current column-patterns are enlarged. Steps (3(a)ii) and (3(a)iii) perform the quantitative control for the size of the column-patterns. In steps (3(a)iv) and (3(a)v), the new indices for enlarging the column-patterns are computed. The actual enlarging of the column-patterns is done in step (3(a)vi). If not all column-patterns satisfy one of the stopping criteria (checked in step (3b)), the current projective approximate inverse with the new projection pattern $\left(J_1^l, \ldots, J_n^l\right)$ is computed in step (3c). If the current step of the $l$-loop is not the last one (checked in step (3d)), the new $J_k^l$-residuals are computed, and the column-patterns are enlarged once more.

In the formulation of the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8), the combined stopping criterion (see dicussion on pages 67–68) for the pattern derivation is used. If only the quantitative stopping criterion for the pattern derivation is desired, step (3(a)i) can be skipped. Alternatively, if only the qualitative stopping criterion is desired, step (3(a)ii) may be replaced by $mt := mfps$.
The length of the $l$-loop in step (3) of algorithm 8 limits the number of pattern enlarging steps, and hence the CPU-time of this algorithm.

Importantly, algorithm 8 offers a large amount of algorithmic parallelism and is thus appealing for parallel computers: the calculation of the projective approximate inverses in steps (1) and (3c) essentially consists in the solution of $n$ independent small linear systems (see algorithm 6). All the $J_k$-residuals in steps (2) and (3d) can be calculated simoultanously for $k = 1, \ldots, n$, and thus in parallel. The entire pattern derivation in step (3a) can be done in parallel for $k = 1, \ldots, n$.
Further, algorithm 8 offers a considerable potentail for vectorization: the matrix-vector multiplications for calculating the $J_k^l$-residuals in steps (2) and (3d), as well as the dot-products in step (3(a)i) for calculating the $Z_L A^{-1}$-norms of the $J_k^l$-residuals, are efficiently vectorizable. Furthermore, the calculation of the projective approximate inverses in steps (1) and (3c) may be – depending on the utilized linear solver – vectorizable.

**Computational Complexity of the `Pattern Adaptive Projective Approximate Inverse` Algorithm**
The computational complexity of the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8) depends on its parameters as well as on the matrices $A$, $Z_L$, $Z_R$ and on the initial projection pattern $(J_1^0, \ldots, J_n^0)$. The steps that contribute to the computational cost are:

**steps** (1)**,** (3c)**:** In these steps, the projective approximate inverse $P^l$ is calculated by invoking algorithm 6 at most $ms + 1$ times with the projection patterns $\left(J_1^l, \ldots, J_n^l\right)$ for $l = 0, \ldots, ms$.

**steps** (2)**,** (3d)**:** Here, the new $J_k^l$-residuals are computed. Theoretically, $2n$ matrix-vector products, namely $P^l\,(Z_R u_k)$, $A\left(P^l Z_R u_k\right)$ and $n$ sums of two vectors of length $n$, i.e. $AP^l Z_R u_k - Z_R u_k$, for $k = 1, \ldots, n$, are necessary. For practical applications, the matrix $Z_R$ should be such that these operations are cheap.
If the Residual-Minimizing smoothed residuals $s_{J_k^l}$ are utilized for the qualitative stopping criterion (see pages 67–68), algorithm 1 is invoked for calculating the Residual-Minimizing smoothed residuals. This requires forming the sum of two (possibly sparse) vectors, two (possibly sparse) dot-products, and one triadic operation (see pages 27–29 for details).

**step** (3(a)i)**:** In this step, either the $Z_L A^{-1}$-norms of $J_k^l$-residuals $r_{J_k}$ or the Euclidian norms of the corresponding Residual-Minimizing smoothed $J_k^l$-residuals $s_{J_k}$ are computed.

**step** (3(a)iv) We consider the computation of the decrease rates $\lambda_j$ in sections 5.2 and 5.3.

Steps (1) and (2) are executed only once, whereas the steps (3(a)i), (3(a)iv), (3c), and (3d) are executed at most $ms$ times for each $k$ with $1 \leq k \leq n$.

**Optimizations for the Practical Implementation**

For the practical implementation of the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8), several optimizations, depending on the projection matrices $Z_L$ and $Z_R$, are possible:

Depending on the matrix $Z_R$, the $J_k^l$-residuals might be handled efficiently as sparse vectors, leading to CPU-time savings when computing the residual norms or the Residual-Minimizing smoothed residuals and its norms in step (3(a)i).

The computational cost of step (3(a)iv) may be reduced by calculating the decrease rates $\lambda_j$ from definition 5.1 not for all elements $j$ of the candidate sets $C_{J_k^l}$, but for some specified subset. Numerous ways of defining such subsets are possible.
For instance, depending on the matrices $A$, $Z_L$ and $Z_R$, it may be possible to find simple criteria for which indices $j \in C_{J_k^l}$ the decrease rates $\lambda_j$ are positive. It suffices to compute only those $\lambda_j$ in step (3(a)iv). Such a criterion is given in lemma 5.6.

CPU-time can possibly be saved by estimating the decrease rates $\lambda_j$ instead of computing them exactly.

The choice of the initial projection pattern $(J_1^0, \ldots, J_n^0)$ has a strong influence on the computational complexity of the algorithm 8. The number of pattern derivation steps, i.e. the length of the $l$-loop in step (3) of the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8) may be reduced by supplying an appropriate initial projection pattern.
Since in practice no information on an appropriate projection pattern for the projective approximate inverse is available, it must be determined adaptively from scratch. In this case, the initial projection pattern has some simple shape. The most obvious initial projection pattern is the diagonal pattern, which furnishes a practical projection method according to definition 4.3 whenever the main diagonals of the matrices $Z_L^H A$ and $Z_L^H Z_R$ are zero-free.

If a sequence of linear systems with a slowly varying coefficient matrix – like a mildly non-linear problem solved with a variant of Newton's method – has to be solved, a once derived projection pattern can be reused as initial pattern for the next linear system. Thereby, the length of the $l$-loop in step (3a) of the above algorithm can be reduced, leading to substantial savings of CPU-time.
Another application with a non-trivial initial projection pattern is at hand, if the convergence of the preconditioned iterative solution process is not satisfactory. In this case, the iteration can be interrupted and the known projection pattern of the approximate inverse can be enlarged further by applying the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8) with the initial projection pattern being the already known projection pattern.

**The Case of diagonal $Z_R$**
We saw in section 4.1 that if the projection matrix $Z_R$ is diagonal, the $J_k$ are the column-patterns of the projective approximate inverse $P$ for $k = 1, \dots, n$, and that the columns of $P$ can be computed completely in parallel by algorithm 7.
In algorithm 9, we give the pseudo-code of the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8) for the case that $Z_R$ is diagonal matrix. The input data and the parameters needed for this algorithm are the same as for algorithm 8. We denote the diagonal elements of the projection matrix $Z_R$ by $d_k$ for $k = 1, \dots, n$. The supplied initial pattern $(J_1^0, \dots, J_n^0)$ must be chosen such that the corresponding projection method is explicit according to definition 4.3. For practical purposes, the projection method on the initial projection pattern should be practical in the sense of definition 4.3, because this guarantees, that the final projection method is practical as well. In this situation, the projective approximate inverse $P$ obtained on output of this algorithm has no vanishing column (see corollary 4.5).
In step (1a) of the `Pattern Adaptive Projective Approximate Inverse` algorithm with diagonal $Z_R$ (algorithm 9), the columns $p_k^0$ of the initial projective approximate inverse $P^0$ are determined. The corresponding $J_k^0$-residuals are calculated in step (1b). The actual pattern derivation is done within the $l$-loop in step (1c). First of all, the qualitative stopping criterion is checked in step (1(c)i). Alternatively to considering the $J_k^l$-residuals $r_{J_k^l}$, the corresponding Residual-Minimizing smoothed residuals $s_{J_k^l}$ can be monitored (see pages 67–68). In step (1(c)ii) the quantitative stopping criterion is checked. If either of the stopping criteria is satisfied, the pattern derivation for this column $p_k$ of the projective approximate inverse $P$ is stopped. In step (1(c)iv) the decrease rates $\lambda_j$ for all elements $j$ of the candidate set $C_{J_k^l}$ are calculated (those quantites are introduced in definition (5.1)), and according to the largest decrease rates, the augmented column-pattern is determined in step (1(c)vi). In step (1(c)vii) the $k$-th column $p_k^l$ of the projective approximate inverse is caculated on the augmented column-pattern. The corresponding new $J_k^l$-residual

1. For $k = 1, \ldots, n$
   (a) Compute the column $p_k^0$ of the projective approximate inverse $P^0$ with the initial column-pattern $J_k^0$ using algorithm 7
   (b) $r_{J_k^0} := d_k \left( A p_k^0 - u_k \right)$
   (c) For $l = 1, \ldots, ms$
      i. If $\left( \left\| r_{J_k^{l-1}} \right\|_{Z_L A^{-1}} < \epsilon_k \right)$ Cycle $k$-loop
      ii. $mt := \min \left( mfps, mf - \left( \# J_k^{l-1} \right) \right)$
      iii. If $(mt == 0)$ Cycle $k$-loop
      iv. Compute the decrease rates $\lambda_j$ for $j$ all elements of the candidate set $C_{J_k^{l-1}}$ (see definition (5.1))
      v. Determine the $mt$ indices $j_1, \ldots, j_{mt}$ according to the largest decrease rates $\lambda_j$
      vi. $J_k^l := J_k^{l-1} \cup (j_1, \ldots, j_{mt})$
      vii. Compute the $k$-th column $p_k^l$ of $P^l$ on the column-pattern $J_k^l$ with algorithm 7
      viii. If $(l \neq ms)$: Compute the new $J_k^l$-residual $r_{J_k^l}$

**Algorithm 9:** `Pattern Adaptive Projective Approximate Inverse` with diagonal $Z_R$

is determined in step (1(c)viii).

Note that the structure of the algorithm 9 is different from the structure of algorithm 8. While in algorithm 8 the $k$-loop (this is step (3a) in algorithm 8 and step (1) in algorithm 9) is contained in the $l$-loop (this is step (3) in algorithm 8 and step (1c) in algorithm 9), in algorithm 9 the $l$-loop is contained in the $k$-loop. Thus, in algorithm 9 the sparsity patterns of the columns of the projective approximate inverse $P$ are determined independently from each other. Conversely, in algorithm 8 the pattern adaption steps must be done syncronously, i.e. the qualitative stopping criterion of the pattern derivation is checked after each pattern adaption step is completed for all columns.

In the above formulation of the `Pattern Adaptive Projective Approximate Inverse` algorithm for diagonal $Z_R$ (algorithm 9), the combined stopping criterion is utilized (see pages 67–68 for a discussion on that matter). If only the quantitative stopping criterion is requested, step (1(c)i) can be skipped. If only the qualitative stopping criterion is desired, step (1(c)ii) may be replaced with $mt := mfps$.

Importantly, the `Pattern Adaptive Projective Approximate Inverse` alogrithm with diagonal $Z_R$ (algorithm 9) is inherently parallel: all steps of the $k$-loop in step 1 of the above algorithm, i.e. the derivation of the column patterns $J_k^l$ and the computation of the corresponding columns of $p_k^l$ of the projective approximate inverse $P^l$, where $0 \leq l \leq ms$, can be done independently from each other, and hence in parallel. Thus, the `Pattern Adaptive Projective Approximate Inverse` algorithm with diagonal $Z_R$ (algorithm 9) is well-suited for parallel computers.
Furthermore, the algorithm 9 offers some potential for vectorization: The matrix-vector products for calculating the $J_k^l$-residuals in steps (1b) and (1(c)viii), and the dot-products for calculating the $Z_L A^{-1}$-norms of the $J_k^l$-residuals in step (1(c)i), are efficiently vectorizable. The potential for vectorization of algorithm 7, which is invoked in steps (1a) and (1(c)vii), depends on the utilized linear solver.

**Computational Complexity of the `Pattern Adaptive Projective Approximate Inverse` Algorithm with diagonal $Z_R$**
The computational complexity of the `Pattern Adaptive Projective Approximate Inverse` algorithm with diagonal $Z_R$ (algorithm 9) depends on its parameters as well as on the matrices $A$, $Z_L$, and on the initial projection pattern $(J_1^0, \ldots, J_n^0)$. The steps that contribute to the computational cost are:

**steps** (1a)**,** (1(c)vii)**:** In these steps, algorithm 7 is invoked at most $ms + 1$ times for calculating the columns $P^l u_k$ of the projective approximate inverses $P^l$ with the column patterns $J_k^l$ for $l = 1, \ldots, ms$ and $k = 1, \ldots, n$.

**steps** (1b)**,** (1(c)viii)**:** Here, the new $J_k^l$-residuals are computed. Theoretically, $n$ sparse scaled matrix-vector products, namely $d_k \left(AP^l u_k\right)$, are necessary.
If the Residual-Minimizing smoothed residuals $s_{J_k^l}$ are utilized for the quantitative stopping criterion of the pattern derivation (see pages 67–68), algorithm 1 is invoked. This involves calculating the sum of two vectors, two dot-products, and one triadic operation (these operations are possibly sparse ).

**step** (1(c)i)**:** In this step, either the $Z_L A^{-1}$-norms of $J_k^l$-residuals $r_{J_k}$ or the Euclidian norms of the corresponding Residual-Minimizing smoothed $J_k^l$-residuals $s_{J_k}$ are computed.

**step** (1(c)iv) We consider the computation of the decrease rates $\lambda_j$ in sections 5.3 and 5.2.

For each $k$ with $1 \leq k \leq n$ the steps (1a) and (1b) are executed only once, whereas the steps (1(c)i), (1(c)iv), (1(c)vii) and (1(c)viii) are executed at most $ms$ times.
For possible optimizations regarding the practical implementation of algorithm 9, the corresponding remarks given to algorithm 8 on pages 72–73 apply.

In the above, we have derived two inherently parallel and vectorizable algorithms for determining projective approximate inverses of non-singular matrices; algorithm 8 for general projection methods and algorithm 9 for projection methods with diagonal $Z_R$. Based on the provided parameters, both algorithms calculate the projective approximate inverse on adaptively determined projection patterns. This involves the decrease rates $\lambda_j$ from definition 5.1.
In the following section, we will consider the actual contour and the practical computation of these decrease rates $\lambda_j$. Further, in section 5.3 consider strategies for estimating the decrease rates $\lambda_j$.

## 5.2 Multivariate Minimizing Pattern Adaption

In this section, we derive an explicit representation of the decrease rates $\lambda_j$ from definition 5.1. Further, we consider the properties of the `Pattern Adaptive Projective Approximate Inverse` algorithms (algorithms 8 and 9) with regards to the decrease rates $\lambda_j$.

**Brief Digest on Least-Squares Problems**
We begin this section with a brief summary of some well-known facts regarding the solution of least-squares problems. For detailed surveys on least-squares problems we refer e.g. to [74] and [75].
Let the matrix $B \in I\!K^{m\times n}$, where $m \geq n$, and let $B$ have full column rank $n$. Let further the vector $b$ be an element of $I\!K^m$. Then the least squares problem

$$\min_{x\in I\!K^n} \left\|Bx - b\right\|_2^2 \tag{5.4}$$

has at least one solution $x_0 \in I\!K^n$. Every solution $x \in I\!K^n$ of (5.4) is a solution of the normal equations

$$B^H Bx = B^H b, \tag{5.5}$$

and vice versa. With the solution $x_0 \in I\!K^n$ of (5.4) the vector $b - Bx_0 \in I\!K^m$ can be formally expressed as

$$b - Bx_0 = Tb, \tag{5.6}$$

where $T$ is an arbitrary projector on the null-space of $B^H$.

We give a technical lemma, see e.g. [32], which we extend to the complex case.

**Lemma 5.2** ([74], pp. 207–216)
*Let the matrix $B \in I\!K^{m\times n}$, where $m \geq n$, and let $B$ have full column rank $n$. Let*

*further $I$ denote the identity matrix in $\mathbb{K}^{m\times m}$. Then the operator $T_B : \mathbb{K}^m \to \mathbb{K}^m$ defined by*

$$T_B := I - B\left(B^H B\right)^{-1} B^H \tag{5.7}$$

*is a Hermitian projector on the null-space of $B^H$.*

**Proof.**

We give the proof for the case $\mathbb{K} = \mathbb{C}$, which proceeds analogously to the proof for the case $\mathbb{K} = \mathbb{R}$ given in [32]:
Obviously, $T_B$ is Hermitian. Because of

$$\begin{aligned} T_B^2 &= (I - B(B^H B)^{-1} B^H)(I - B(B^H B)^{-1} B^H) \\ &= I - 2(I - B(B^H B)^{-1} B^H) + B(B^H B)^{-1} \underbrace{B^H B(B^H B)^{-1}}_{=I} B^H \\ &= T_B \end{aligned}$$

$T_B$ is a projector. And because of

$$B^H T_B = B^H - \underbrace{B^H B\left(B^H B\right)^{-1}}_{=I} B^H = 0$$

$T_B$ projects on the null-space of $B^H$. $\diamondsuit$

The following lemma was originally given by Cosgrove, Diaz and Griewank in [17] for real matrices. A detailed discussion of this statement is given by Gould and Scott in [32] on pages 608–610. We extend it to the complex case.

**Lemma 5.3** (Augmented Least Squares, [17], p. 101)
*Let the matrix $A \in \mathbb{K}^{n\times n}$ be non-singular, and let the vector $d \in \mathbb{K}^n$ be arbitrary non-vanishing. Let $J = (j_1, \ldots, j_k)$, with $\#J = k < n$, be the pattern of the vector $x_0 \in \mathbb{K}^n$, where the vector $x_0$ is the solution of the least squares problem $\min_{w\in\mathbb{K}^n_J} \left\|A(.,J)\,w - d\right\|_2^2$, and let the vector $r_0 \in \mathbb{K}^n$ be defined by $r_0 := A(.,J)\,x_0 - d$. Let further $T_{A(.,J)}$ denote the Hermitian projector on the null-space of $(A(.,J))^H$ from equation (5.7). Then for each $1 \le j \le n$ with $j \notin J$ the equation*

$$\min_{w\in\mathbb{K}^n_{J\cup(j)}} \left\|A(.,J\cup(j))\,w - d\right\|_2^2 = \left\|r_0\right\|_2^2 - \frac{\mid u_j^H A^H r_0 \mid^2}{\left\|T_{A(.,J)} A u_j\right\|_2^2}, \tag{5.8}$$

*holds.*

**Proof.**

The proof for the case $I\!K = I\!R$ is given in [17]. The following proof for the complex case proceeds analogously:
For notational convenience we set $A_0 := A(.,J)$, $A_1 := A(.,J \cup (j))$ and $a_j := Au_j$.
First of all, we show that the fraction on the right-hand side of equation (5.8) is well-defined:
It suffices to show, that the denominator $\left\|T_{A_0} a_j\right\|_2^2$ is positive. Since the matrix $A$ is non-singular, we know that the $j$-th column $a_j$ of $A$ is not an element of the range of the matrix $A_0$, and thus we have

$$T_{A_0} a_j = a_j - A_0 \left(\left(A_0^H A_0\right)^{-1} A_0^H a_j\right) \neq 0.$$

Hence we have

$$\left\|T_{A_0} a_j\right\|_2^2 > 0.$$

We prove that the operator

$$T_{A_1} := T_{A_0} - \frac{T_{A_0} a_j a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^2} \tag{5.9}$$

is Hermitian projector on the null-space of $A_1^H$:

i) $T_{A_1}$ is a projector:

$$\begin{aligned}
T_{A_1}^2 &= T_{A_0}^2 - 2T_{A_0} \frac{T_{A_0} a_j a_j^H T_B}{\left\|T_{A_0} a_j\right\|_2^2} + \frac{T_{A_0} a_j a_j^H T_{A_0} T_{A_0} a_j a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^4} \\
&= T_{A_0} - 2\frac{T_{A_0} a_j a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^2} + \frac{T_{A_0} a_j a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^2} \\
&\quad \text{since } \ a_j^H T_{A_0}^2 a_j = \left\|T_{A_0} a_j\right\|_2^2 \\
&= T_{A_1}.
\end{aligned}$$

ii) $T_{A_1}$ projects on the null-space of $A_1^H$:
We have

$$\begin{aligned}
A_0^H T_{A_1} &= A_0^H \left(T_{A_0} - \frac{T_{A_0} a_j a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^2}\right) \\
&= 0 - \frac{A_0^H T_{A_0} a_j a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^2} \\
&= 0,
\end{aligned}$$

since $T_{A_0}$ is a projector on the null-space of $A_0^H$, and

$$\begin{aligned} a_j^H T_{A_1} &= a_j^H \left( T_{A_0} - \frac{T_{A_0} a_j a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^2} \right) \\ &= a_j^H T_{A_0} - \left\|T_{A_0} a_j\right\|_2^2 \frac{a_j^H T_{A_0}}{\left\|T_{A_0} a_j\right\|_2^2} \\ &= 0, \end{aligned}$$

and thus

$$A_1^H T_{A_1} = \begin{pmatrix} A_0^H \\ a_j^H \end{pmatrix} T_{A_1} = 0.$$

We verify equation (5.8):

$$\begin{aligned} \min_{w \in \mathbb{C}^n_{J \cup (j)}} \left\|A_1 w - d\right\|_2^2 = & \left\|T_{A_1} d\right\|_2^2 \\ & \text{by equation (5.6)} \\ = & \; d^H T_{A_1}^2 d \\ = & \; d T_{A_1} d \\ & \text{since } T_{A_1} \text{ is a projector} \\ = & \; d^H T_{A_0} d - \frac{d^H T_{A_0} a_j a_j^H T_{A_0} d}{\left\|T_{A_0} a_j\right\|_2^2} \\ & \text{by (5.9)} \\ = & \left\|T_{A_0} d\right\|_2^2 - \frac{\mid a_j^H T_{A_0} d \mid^2}{\left\|T_{A_0} a_j\right\|_2^2} \\ = & \left\|r_0\right\|_2^2 - \frac{\mid a_j^H r_0 \mid^2}{\left\|T_{A_0} a_j\right\|_2^2}, \end{aligned}$$

by equation (5.6), and the proof is complete. ◊

The above lemma gives a statement on the change of the norm of a least-squares solution, if one column is added to the matrix.
With the following lemma, we establish a relation between least-squares problems and the $Z_L A^{-1}$-norm minimization problems which occur in the definition of the decrease rates $\lambda_j$ (see definition 5.1).

**Lemma 5.4**
*Let $Z_L A^{-1} \in \mathbb{K}^{n \times n}$ be Hermitian positive definite and let $L^H L$ be the Cholesky-factorization of $Z_L A^{-1}$. Then, for $x \in \mathbb{K}^n$ arbitrary, the following identity holds:*

$$\left\|x\right\|_{Z_L A^{-1}} = \left\|Lx\right\|_2. \tag{5.10}$$

**Proof.**

The assertion follows directly from

$$\left\|x\right\|^2_{Z_L^H A^{-1}} = x^H Z_L^H A^{-1} x = x^H L^H L x = \left\|Lx\right\|_2^2 .$$

$\diamondsuit$

Now all tools for giving the explicit representation of the decrease rates $\lambda_j$ from definition 5.1 are at hand. By combining lemmata 5.3 and 5.4 we obtain the following new theorem, which states an explicit representation of the decrease rates $\lambda_j$ from definition 5.1:

**Theorem 5.5** (Multivariate Minimizing Pattern Adaption)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in I\!K^{n\times n}$ on the projection pattern $(J_1,\ldots,J_n)$. Let the matrix $Z_L A^{-1} \in I\!K^{n\times n}$ be Hermitian positive definite. Then the candidate sets $C_{J_k}$ from definition 5.1 have the form*

$$C_{J_k} = \{j \mid j \notin J_k\} \tag{5.11}$$

*for $k = 1,\ldots,n$, and the decrease rates $\lambda_j$ from definition* 5.1 *can be written as*

$$\lambda_j = \frac{\mid u_j^H Z_L^H r_{J_k} \mid^2}{\left\|u_j\right\|^2_{Z_L^H A} - g_j^H y_j}, \tag{5.12}$$

*where the vector $g_j \in I\!K^{\#J_k}$ is defined by*

$$g_j := \left(Z_L^H A\right)(J_k, j), \tag{5.13}$$

*and the vector $y_j \in I\!K^{\#J_k}$ is the solution of the $(\#J_k \times \#J_k)$-linear system*

$$\left(Z_L^H A\right)(J_k, J_k) y_j = g_j. \tag{5.14}$$

**Proof.**

We give the proof for an arbitrary integer $k$ with $1 \le k \le n$. For notational convenience, we set $A_0 = A(.,J_k)$, $A_1 = A(.,J_k \cup (j))$ and $a_j = Au_j$.
First of all, we consider the candidate set $C_{J_k}$ from equation (5.11):

Since the matrix $Z_L A^{-1}$ is Hermitian positive definite, the matrix $Z_L^H A$ is Hermitian positive definite as well. Thus, all $(J,J)$-reduced matrices $\left(Z_L^H A\right)(J,J)$ are non-singular for any column pattern $J$. Hence, the candidate set $C_{J_k}$ contains all indices $j$ which are not elements of $J_k$.
We verify assertion (5.12):
We have

$$\begin{aligned}
\left\|r_{J_k\cup(j)}\right\|^2_{Z_L A^{-1}} &= \min_{w\in \mathbb{K}^n_{J_k\cup(j)}} \left\|Aw - Z_R u_k\right\|^2_{Z_L A^{-1}} \\
&\quad \text{by theorem 4.9} \\
&= \min_{w\in \mathbb{K}^n_{J_k\cup(j)}} \left\|LAw - LZ_R u_k\right\|^2_2 \qquad (5.15) \\
&\quad \text{by lemma 5.4,}
\end{aligned}$$

where $L^H L$ is the Cholesky-decomposition of $Z_L A^{-1}$.

We apply lemma 5.3 with $LA$ instead of $A$, $Z_R u_k$ instead of $d$ to the least-squares problem in equation (5.15), and we obtain

$$\min_{w\in \mathbb{K}^n_{J_k\cup(j)}} \left\|LAw - LZ_R u_k\right\|^2_2 = \min_{w\in \mathbb{K}^n_{J_k}} \left\|LAw - LZ_R u_k\right\|^2_2 - \frac{\mid u_j^H A^H L^H L r_{J_k} \mid^2}{\left\|T_{LA_0} L a_j\right\|^2_2},$$

where the Hermitian projector $T_{LA_0}$ on the null-space of $(LA_0)^H$ from lemma 5.2 has the form:

$$T_{LA_0} = \; I - LA_0 \left[(LA_0)^H LA_0\right]^{-1} (LA_0)^H . \qquad (5.16)$$

Thus, with (5.15) and by noting that

$$\begin{aligned}
\min_{w\in \mathbb{K}^n_{J_k}} \left\|LAw - LZ_R u_k\right\|^2_2 &= \min_{w\in \mathbb{K}^n_{J_k}} \left\|LAw - LZ_R u_k\right\|^2_2 \\
&= \min_{w\in \mathbb{K}^n_{J_k}} \left\|Aw - Z_R u_k\right\|^2_{Z_L A^{-1}} \\
&= \left\|r_{J_k}\right\|^2_{Z_L A^{-1}},
\end{aligned}$$

with theorem 4.9, it suffices to show that the fraction

$$\frac{\mid u_j^H A^H L^H L r_{J_k} \mid^2}{\left\|T_{LA_0} L a_j\right\|^2_2} \qquad (5.17)$$

equals the fraction in equation (5.12).
Since the matrix $Z_L A^{-1}$ is Hermitian, we have

$$A^H L^H L = A^H Z_L A^{-1} = A^H \left(Z_L A^{-1}\right)^H = A^H A^{-H} Z_L^H = Z_L^H,$$

and thus we have

$$u_j^H A^H L^H L r_{J_k} = u_j^H Z_L^H r_{J_k},$$

proving the equality of the nominators of the fractions (5.12) and (5.17).
It remains to show that the denominators of the fractions (5.12) and (5.17) are equal. We consider the denominator of the fraction in equation (5.17): The matrix in the brackets equation of (5.16) has the form:

$$\begin{aligned}
(LA_0)^H LA_0 &= A_0^H L^H LA_0 \\
&= A_0^H Z_L A^{-1} A_0 \\
&= \left(u_{j_1}, \ldots, u_{j_{\#J_k}}\right)^H A^H Z_L \left(u_{j_1}, \ldots, u_{j_{\#J_k}}\right) \\
&\quad \text{since } A_0 := A(., J_k) = A \cdot \left(u_{j_1}, \ldots, u_{j_{\#J_k}}\right) \\
&= \left(u_{j_1}, \ldots, u_{j_{\#J_k}}\right)^H Z_L^H A \left(u_{j_1}, \ldots, u_{j_{\#J_k}}\right) \\
&\quad \text{since } A^H Z_L \text{ is Hermitian positive definite} \\
&= \left(Z_L^H A\right)(J_k, J_k).
\end{aligned}$$

For notational convenience, we define the $(\#J_k \times \#J_k)$-matrix $B_{J_k}$ by

$$B_{J_k} := \left(Z_L^H A\right)(J_k, J_k). \tag{5.18}$$

Observe that

$$\begin{aligned}
A_0^H L^H La_j &= \left(u_{j_1}, \ldots, u_{j_{\#J_k}}\right)^H A^H Z_L A^{-1} A u_j \\
&\quad \text{since } Z_L A^{-1} \text{ is Hermitian} \\
&= \left(u_{j_1}, \ldots, u_{j_{\#J_k}}\right)^H A^H A^{-H} Z_L^H A u_j \\
&= \left(Z_L^H A\right)(J_k, j) \\
&= g_j,
\end{aligned} \tag{5.19}$$

where $g_j$ is the vector from equation (5.13), and further

$$\begin{aligned}
a_j^H L^H La_j &= u_j^H A^H Z_L A^{-1} A u_j \\
&\quad \text{since } Z_L A^{-1} \text{ is Hermitian} \\
&= u_j^H A^H A^{-H} Z_L^H A u_j \\
&= u_j^H Z_L^H A u_j \\
&= \left\|u_i\right\|_{Z_L^H A}^2 .
\end{aligned} \tag{5.20}$$

By putting it all together we have

$$\begin{aligned}\left\|T_{LA_0}La_j\right\|_2^2 &= a_j^H L^H T_{LA_0}^H T_{LA_0} La_j \\ &= a_j^H L^H T_{LA_0} La_j \\ &\quad \text{since } T_{LA_0} \text{ is a Hermitian projector} \\ &= a_j^H L^H La_j - a_j^H L^H LA_0 B_{J_k}^{-1} A_0^H L^H La_j \\ &\quad \text{by (5.16) and (5.18)} \\ &= \left\|u_i\right\|_{Z_L^H A}^2 - g_j^H B_{J_k}^{-1} g_j \\ &\quad \text{by (5.19) and (5.20)} \\ &= \left\|u_i\right\|_{Z_L^H A}^2 - g_j^H y_j,\end{aligned}$$

with (5.14), and the proof is complete.

$\diamondsuit$

With the explicit representation of the decrease rates $\lambda_j$ from (5.12), the impact of the $J_k$-residual norms caused by augmenting the column patterns is known:

$$\left\|r_{J_k\cup(j)}\right\|_{Z_L A^{-1}}^2 = \left\|r_{J_k}\right\|_{Z_L A^{-1}}^2 - \frac{\mid u_j^H Z_L^H r_{J_k} \mid^2}{\left\|u_j\right\|_{Z_L^H A}^2 - g_j^H y_j}, \tag{5.21}$$

where the vectors $g$ and $y$ are from (5.13) and (5.14). With the identity $\left\|r_{J_k}\right\|_{Z_L A^{-1}}^2 = \left\|e_{J_k}\right\|_{A^H Z_L}^2$ the analogous statement

$$\left\|e_{J_k\cup(j)}\right\|_{Z_L^H A}^2 = \left\|e_{J_k}\right\|_{Z_L^H A}^2 - \frac{\mid u_j^H Z_L^H r_{J_k} \mid^2}{\left\|u_j\right\|_{Z_L^H A}^2 - g_j^H y_j} \tag{5.22}$$

is obtained for the corresponding $J_k$-errors.

Since the decrease rates $\lambda_j$ are positive only if the nominator $u_j^H Z_L^H r_{J_k}$ in (5.12) is non-vanishing, for the implementation of the `Pattern Adaptive Projective Approximate Inverse` algorithms (algorithms 8 and 9) it is sufficient to compute the decrease rates $\lambda_j$ only for $j \in L_{J_k}$ with

$$L_{J_k} := \left\{ j \mid \left(Z_L^H r_{J_k}\right)_j \neq 0,\ j \notin J_k \right\} \tag{5.23}$$

for $k = 1, \ldots, n$.
The following lemma states that the sets $L_{J_k}$ defined in (5.23) are non-empty unless the corresponding $J_k$-residual is vanishing:

**Lemma 5.6**
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L$, $Z_R \in I\!K^{n\times n}$ on the projection pattern $(J_1, \ldots, J_n)$ and let the product matrix $Z_L^H A$ be Hermitian positive definite. Let further $1 \leq k \leq n$ be arbitrary such that the $J_k$-residual $r_{J_k}$ is non-vanishing. Then the set $L_{J_k}$ defined in (5.23) is a non-empty subset of the candidate set $C_{J_k}$ from definition 5.1, i.e. there exists at least one index $j \in L_{J_k} \subset C_{J_k}$ with*

$$\left(Z_L^H r_{J_k}\right)_j \neq 0. \tag{5.24}$$

**Proof.**

We prove assertions for one fixed number $k$ with $1 \leq k \leq n$:
The inclusion $L_{J_k} \subset C_{J_k}$ is trivial, since for Hermitian positive definite $Z_L^H A$, the candidate set $C_{J_k}$ equals $\{1, \ldots, n\} \setminus J_k$.
We verify assertion (5.24):
By lemma 4.8, we know that

$$j \in J_k \Longrightarrow \quad \left(Z_L^H r_{J_k}\right)_j = 0.$$

We verify the assertion by contradiction:
If the set $L_{J_k}$ was empty, we would have $Z_L^H r_{J_k} = (0, \ldots, 0)^H$, which would imply that the matrix $Z_L$ was singular, since the $J_k$-residual $r_{J_k}$ is non-vanishing.

$\diamondsuit$

Altogether, for projection methods with Hermitian positive definite $Z_L^H A$ we have the following results:

i) By lemma 4.4, the corresponding projective approximate inverse $P$ is – independently of the chosen projection pattern – well defined, and $P$ has always an unique explicit representation.

ii) The quality of the approximation of the projective approximate inverse $P$ to the true inverse $A^{-1}$ is stated by theorem 4.9.

iii) Unless the $J_k$-residuals are vanishing, the pattern derivation process will determine indices which can be added to the current column-pattern $J_k$ in such a way that the corresponding $Z_L A^{-1}$-norm of the residual deceases.

These favourable properties need not hold true for projection methods with non-Hermitian or indefinite $Z_L^H A$. For such projection methods, it depends on the particular properties of the matrix $A$, as well as on the choice of the projection matrices $Z_L$ and $Z_R$, in how far any of the properties from the above list apply. Nevertheless, our numerical experiments indicate that, although in many cases not covered by theory, particular projection methods are competative to some of the state-of-the-art preconditioning techniques (see chapter 7 for an ample discussion).

**The `Multivariate Pattern Adaptive Projective Approximate Inverse` Algorithm**

We give a pseudo-code form of the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8) for projection methods, which includes the explicit representation of the decrease rates from theorem 5.5. This algorithm differs from the `Pattern Adaptive Projective Approximate Inverse` algorithm only in step (3(a)iv), which is concerned with the determination of the decrease rates $\lambda_j$ from definition 5.3. The input data and the parameters for the resulting `Multivariate Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 10) are identical with those for algorithm 8 (see the discussion on pages 67–73). Note that the decrease rates $\lambda_j$ from (5.12) are well-defined only if the product matrix $Z_L^H A$ is Hermitian positive definite. On output, the `Multivariate Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 10) returns a projective approximate inverse $P$ on an adaptively derived projection pattern.

The `Multivariate Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 10) is identical with algorithm 8 except of the computation of the decrease rates in step (3(a)iv). For a discussion on the mode of operation and on control strategies for algorithm 10 see the discussion on pages 67–73.

**The Computational Cost of the Decrease Rates**

In step (3(a)iv) of the `Multivariate Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 10), only positive decrease rates $\lambda_j$ are computed by determining the sets $L_{J_k^{l-1}}$ from (5.23) in advance. This involves the sparse dot-products $u_j^H Z_L^H r_{J_k^{l-1}}$ for $j \notin J_k^{l-1}$. For determining the vectors $g_j$, the sparse matrix-vector product $\left(Z_L^H A\right)(J_k, j)$ is formed. The matrix $Z_L$ should have a special form, such that this operation is computationally cheap. The matrix

$$\left(Z_L^H A\right)(J_k, J_k)$$

is already known from steps (1) or (3c). Since the linear systems

$$\left(Z_L^H A\right)(J_k, J_k)\, y_j = g_j \tag{5.25}$$

1. Compute the projective approximate inverse $P^0$ with the initial projection pattern $(J_1^0, \ldots, J_n^0)$ using algorithm 6
2. $r_{J_k^0} := (AP^0 - I)\, Z_R u_k$ for $k = 1, \ldots, n$
3. For $l = 1, \ldots, ms$
   (a) For $k = 1, \ldots, n$
      i. If $\left( \left\| r_{J_k^{l-1}} \right\|_{Z_L A^{-1}} < \epsilon_k \right)$ Cycle $k$-loop
      ii. $mt := \min\left(mfps, mf - \left(\# J_k^{l-1}\right)\right)$
      iii. If $(mt == 0)$ Cycle $k$-loop
      iv. Determine the set $L_{J_k^{l-1}}$ as defined in (5.23), and determine for all $j \in L_{J_k^{l-1}}$ the vector $g_j := \left(Z_L^H A\right)(J_k, j)$, the solution $y_j$ of the $(\# J_k \times \# J_k)$-linear system
      $$\left(Z_L^H A\right)(J_k, J_k)\, y_j = g_j$$
      and compute the decrease rates
      $$\lambda_j := \frac{\mid \left(Z_L^H r_{J_k}\right)_j \mid^2}{\left\| u_j \right\|^2_{A^H Z_L} - g_j^H y_j}$$
      v. Determine the $mt$ indices $j_1, \ldots, j_{mt}$ according to the largest decrease rates $\lambda_j$
      vi. $J_k^l := J_k^{l-1} \cup (j_1, \ldots, j_{mt})$
   (b) If no column-pattern was enlarged in step (3a) : STOP
   (c) Compute the approximate inverse $P^l$ with the projection pattern $\left(J_1^l, \ldots, J_n^l\right)$ using algorithm 6
   (d) If $(l \neq ms)$: Compute the new $J_k^l$-residuals $r_{J_k^l}$ for $k = 1, \ldots, n$

**Algorithm 10:** `Multivariate Pattern Adaptive Projective Approximate Inverse`

are solved for many right-hand sides, special solvers for these linear systems should be applied. For instance, if the $QR$-decomposition of the matrix

$$\left(Z_L^H A\right)(J_k, J_k)$$

is known from steps (1) or (3c), the linear systems (5.25) can efficiently be solved as

$$y_j = R^{-1}Q^H g_j$$

by backward substitution. Alternatively, iterative methods for multiple right-hand-sides can be applied.

As discussed for algorithm 8 (see page 71), the `Multivariate Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 10) is inherently parallel and vectorizable, and thus appealing for the implementation on vector and parallel computers.
Additionally, since the decrease rates $\lambda_j$ in step (3(a)iv) of the above algorithm can be calculated independently from each other, this step yields a considerable potential for vectorization and parallelization.

**Optimizations for the Practical Implementation**
The remarks given on pages 72–73 regarding possible optimizations of algorithm 8 apply.

If only one new index $j$ is added to the augmented pattern in each step, i.e. $mt = 1$, the Euclidian norms of the $J_k^l$-residual $r_{J_k^l}$ may be updated by

$$||r_{J_k^l}||_2^2 := ||r_{J_k^{l-1}}||_2^2 - \lambda_j$$

rather than directly computed.

If more than one pattern derivation step is made for each column of the projective approximate inverse, i.e. if the parameter $ms$ is larger than one, the quantities $\left\|u_j\right\|^2_{A^H Z_L}$ in the denominator of the decrease rates in step (3(a)iv) may be stored in a supplemental vector after being initially determined.

The computational cost of the pattern derivation process may be reduced by computing the decrease rates $\lambda_j$ only for some subset $\tilde{L}_{J_k}$ of the set $L_{J_k}$ from (5.23). In theory, any heuristic strategy for defining such a subset $\tilde{L}_{J_k}$ is possible. For instance, by prescribing some positive threshold $\mu \in I\!R$, the set $\tilde{L}_{J_k}$ can be defined by monitoring the size of the nominators of the decrease rates from (5.12):

$$\tilde{L}_{J_k} := \left\{ i \in L_{J_k} \mid \left| \left( Z_L^H r_{J_k} \right)_i \right| > \mu \right\}. \tag{5.26}$$

We note that the quality of the projective approximate inverse determined by such a truncated pattern derivation process might suffer.

**The Case of Diagonal $Z_R$**
For projection methods with a diagonal right-hand side projection matrix $Z_R$, the projective approximate inverse $P$ can be calculated with algorithm 9, where the actual calculation of the decrease rates $\lambda_j$ in step (1(c)iv) of algorithm 9 is done exactly in the same way as in step (3(a)iv) of algorithm 10. With the resulting algorithm, all columns $p_k$ of the projective approximate inverse $P$, as well as the corresponding sparsity patterns, can be determined independently from each other for $k = 1, \ldots, n$.

In this section, we have derived a new explicit representation of the decrease rates $\lambda_j$ from definition 5.1 for general projection methods. We have considered algorithms 8 and 9 including the explicit calculation of these decrease rates and we have discussed the computational cost caused by the decrease rates.
In the following section, we consider estimates for the decrease rates $\lambda_j$, which possibly are cheaper to compute.

## 5.3 Univariate Minimizing Pattern Adaption

In this section, we introduce a strategy for estimating the decrease rates $\lambda_j$ from definition 5.3. The basic idea for estimating the decrease rates presented in this section has been introduced by Cosgrove, Diaz and Griewank in [17] for the Frobenius norm minimizing preconditioning technique, which has been classified as a particular projection method by Zimmermann in [83] (see section 6.2). We extend the approach of Cosgrove, Diaz and Griewank to general projection methods.

**The Basic Concept for Estimating the Decrease Rates**
Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n \times n}$ determined by an explicit projection method according to definition 4.3 with the projection matrices $Z_L,\ Z_R \in I\!K^{n \times n}$ on the projection pattern $(J_1, \ldots, J_n)$. Let further $p_k$ denote the $k$-th column of $P$ for $k = 1, \ldots n$.
For Hermitian positive definite $Z_L A^{-1}$, the decrease rates $\lambda_j$ from definition 5.1 are linked by (4.19) to a multidimensional minimization problem:

$$\begin{aligned} \lambda_j &= \left\| r_{J_k} \right\|^2_{Z_L A^{-1}} - \min_{w \in I\!K^n_{J_k \cup (j)}} \left\| Aw - Z_R u_k \right\|^2_{Z_L A^{-1}} \\ &= \left\| r_{J_k} \right\|^2_{Z_L A^{-1}} - \min_{\substack{v \in I\!K^n_{J_k} \\ \zeta \in I\!K}} \left\| Av + \zeta A u_j - Z_R u_k \right\|^2_{Z_L A^{-1}} \end{aligned} \tag{5.27}$$

The strategy of estimating these decrease rates considered in this section is essentially based on substituting the vector $v$ in the multidimensional minimization problem (5.27) by the fixed vector $p_k$, the $k$-th column of the projective approximate inverse $P$ determined on the current projection pattern. This strategy has been proposed by Cosgrove, Diaz and Griewank in [17] for the Frobenius norm minimizing preconditioning technique (see section 6.2). The advantage of this approach is that for calculating the estimated decrease rates only a one-dimensional minimization problem is considered instead of the multidimensional minimization problem (5.27). We give the definition of the estimated decrease rates:

**Definition 5.7** (Univariate Decrease Rate)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in I\!K^{n\times n}$ on the projection pattern $(J_1,\dots,J_n)$. Let this projection method be $J_k$-explicit for some $k \in I\!N$ according to definition 4.3. Let $C_{J_k}$ denote the candidate set*

$$C_{J_k} := \left\{j \mid j \notin J_k,\ \left(Z_L^H A\right)(J_k \cup (j)\,, J_k \cup (j)) \text{ is non-singular}\right\} \tag{5.28}$$

*of the column-pattern $J_k$ from definition 5.1 Then for all $j \in C_{J_k}$ we call the non-negative real number $\theta_j$ defined by*

$$\theta_j := \left\|r_{J_k}\right\|^2_{Z_L A^{-1}} - \min_{\zeta\in I\!K} \left\|r_{J_k} - \zeta A u_j\right\|^2_{Z_L A^{-1}} \tag{5.29}$$

**univariate decrease rate** *of the $J_k$-residual $r_{J_k}$ for the index $j$.*

Obviously, if the length $\#J_k$ of the column-pattern $J_k$ is larger than one, then solving the one-dimensional minimization problem (5.29) is computationally cheaper than solving the $(\#J_k + 1)$-dimensional minimization problem (5.27). Thus, utilizing the univariate decrease rates $\theta_j$ from definition 5.7 for the pattern derivation instead of the decrease rates $\lambda_j$ from definition 5.1 may be advantageous. A comparison of both strategies for the Frobenius norm minimizing approach is given by Gould and Scott in [32]. We survey this matter in chapter 7 for several projection methods.
The following lemma gives an explicit representation of the univariate decrease rates from definition 5.7:

**Lemma 5.8** (Explicit Calculation of the Univariate Decrease Rates)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in I\!K^{n\times n}$ on the projection pattern $(J_1,\dots,J_n)$.*

*i) Let $I\!K = I\!R$ and let the projection method be $J_k$-explicit for some $1 \le k \le n$ according to definition 4.3. Let the matrix $Z_L A^{-1} \in I\!R^{n\times n}$ be positive real and let $M$ denote the symmetric part of $Z_L A^{-1}$, then the univariate decrease rates $\theta_j$ from definition 5.7 have the form*

$$\theta_j = \frac{\left(u_j^T A^T M r_{J_k}\right)^2}{\left\|u_j\right\|^2_{Z_L^T A}}, \tag{5.30}$$

*for all elements $j$ of the candidate set $C_{J_k}$ from (5.28).*

*ii) Let $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ and let the matrix $Z_L A^{-1} \in \mathbb{K}^{n\times n}$ be Hermitian positive definite, then the univariate decrease rates $\theta_j$ from definition* 5.7 *are*

$$\theta_j = \frac{\mid u_j^H Z_L^H r_{J_k} \mid^2}{\left\|u_j\right\|_{Z_L^H A}^2}, \tag{5.31}$$

*for $1 \leq j \leq n$ with $j \notin J_k$ and for $k = 1, \ldots, n$.*

**Proof.**

First, we prove assertion $i$):
The function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f\left(\zeta\right) := \zeta^2 \left\|Au_j\right\|_M^2 + 2\zeta u_j^T A^T M r_{J_k} + \left\|r_{J_k}\right\|_M^2$$

satisfies $f(\zeta) = \left\|r_{J_k} + \zeta Au_j\right\|_M^2$. The first derivative of $f$ is

$$f'(\zeta) = 2\zeta \left\|Au_j\right\|_M^2 + 2u_j^T A^T M r_{J_k},$$

and we have

$$\begin{aligned} & f'(\zeta_0) && = 0 \\ \iff \quad & \zeta_0 && = -\frac{u_j^T A^T M r_{J_k}}{\left\|Au_j\right\|_M^2}. \end{aligned}$$

Since the second derivative $f''(\zeta) = 2\left\|Au_j\right\|_M^2$ is positive (since $Au_j \neq (0, \ldots, 0)^T$), we know that the function $f$ has an absolute minimum at the point $\zeta_0$. We evaluate $f(\zeta_0)$:

$$\begin{aligned} f(\zeta_0) =& \frac{\left(u_j^T A^T M r_{J_k}\right)^2}{\left\|Au_j\right\|_M^2} - 2\frac{\left(u_j^T A^T M r_{J_k}\right)^2}{\left\|Au_j\right\|_M^2} + \left\|r_{J_k}\right\|_M^2 \\ =& \left\|r_{J_k}\right\|_M^2 - \frac{\left(u_j^T A^T M^T r_{J_k}\right)^2}{\left\|Au_j\right\|_M^2}, \end{aligned}$$

and with $\left\|Au_j\right\|_M^2 = \left\|u_j\right\|_{Z_L^T A}^2$ the proof of assertion $i$) is complete.
We verify assertion $ii$) for $1 \leq k \leq n$ arbitrary fixed:

We consider the function $f : I\!\!K \to I\!\!R$ defined by

$$
\begin{aligned}
f(\zeta) &:= \left\| r_{J_k} + \zeta A u_j \right\|^2_{Z_L A^{-1}} \\
&= r^H_{J_k} Z_L A^{-1} r_{J_k} + \zeta r^H_{J_k} Z_L A^{-1} A u_j \\
&\quad + \bar{\zeta} u^H_j A^H Z_L A^{-1} r_{J_k} + \zeta\bar{\zeta} u^H_j A^H Z_L A^{-1} A u_j \\
&= \left\| r_{J_k} \right\|^2_{Z_L A^{-1}} + \zeta r^H_{J_k} Z_L u_j + \bar{\zeta} u^H_j Z^H_L r_{J_k} + |\zeta|^2 \left\| A u_j \right\|^2_{Z_L A^{-1}} \\
&\quad \text{since } Z_L A^{-1} \text{ is Hermitian positive defnite} \\
&= |\ \zeta\ |^2 \left\| A u_j \right\|^2_{Z_L A^{-1}} + 2\mathrm{Re}\left( \zeta u^H_j Z^H_L r_{J_k} \right) + \left\| r_{J_k} \right\|^2_{Z_L A^{-1}} \\
&= \left( \zeta_1^2 + \zeta_2^2 \right) \left\| A u_j \right\|^2_{Z_L A^{-1}} + 2(\zeta_1\gamma_1 - \zeta_2\gamma_2) + \left\| r_{J_k} \right\|^2_{Z_L A^{-1}},
\end{aligned}
$$

where $\zeta = \zeta_1 + i\zeta_2$ and $u^H_j Z^H_L r_{J_k} = \gamma_1 + i\gamma_2$.
For minimizing the function $f$, we must find out where its first derivatives vanish:

$$
\begin{aligned}
\left( \frac{\partial f}{\partial \zeta_1}, \frac{\partial f}{\partial \zeta_2} \right) &= \left( 2\zeta_1 \left\| A u_j \right\|^2_{Z_L A^{-1}} + 2\gamma_1, 2\zeta_2 \left\| A u_j \right\|^2_{Z_L A^{-1}} - 2\gamma_2 \right) \\
&= (0,0) \\
\iff \qquad \zeta_1 &= -\frac{\gamma_1}{\left\| A u_j \right\|^2_{Z_L A^{-1}}} \text{ and } \zeta_2 = \frac{\gamma_2}{\left\| A u_j \right\|^2_{Z_L A^{-1}}}.
\end{aligned}
$$

Since the Hessian matrix of the second derivatives of $f$ equals

$$
\begin{pmatrix} 2\left\| A u_j \right\|^2_{Z_L A^{-1}} & 0 \\ 0 & 2\left\| A u_j \right\|^2_{Z_L A^{-1}} \end{pmatrix},
$$

and is thus symmetric positive definite, we know that the function $f$ has an absolute minimum at the point

$$
\zeta_0 = \left\| A u_j \right\|^{-2}_{Z_L A^{-1}} \left( -\gamma_1 + i \cdot \gamma_2 \right).
$$

By evaluating $f$ at the point $\zeta_0$ we have

$$
\begin{aligned}
f(\zeta_0) &= \frac{\gamma_1^2 + \gamma_2^2}{\left\| A u_j \right\|^2_{Z_L A^{-1}}} - 2\frac{\gamma_1^2 + \gamma_2^2}{\left\| A u_j \right\|^2_{Z_L A^{-1}}} + \left\| r_{J_k} \right\|^2_{Z_L A^{-1}} \\
&= \left\| r_{J_k} \right\|^2_{Z_L A^{-1}} - \frac{|\ u^H_j Z^H_L r_{J_k}\ |^2}{\left\| A u_j \right\|^2_{Z_L A^{-1}}},
\end{aligned}
$$

and with $\left\| A u_j \right\|^2_{Z_L A^{-1}} = \left\| u_j \right\|^2_{Z^H_L A}$ the proof of assertion $ii)$ is complete.

$\diamondsuit$

For Hermitian positive definite $Z_L A^{-1} \in \mathbb{K}^{n\times n}$, the univariate decrease rates $\theta_j$ from definition 5.7 with the explicit representation given in lemma 5.8 can be utilized for the pattern derivation process.
For positive real $Z_L A^{-1} \in \mathbb{R}^{n\times n}$, further considerations for the practical application are necessary:

**The Univariate Decrease Rates in the Positive Real Case**
For projection methods where the product matrix $Z_L A^{-1} \in \mathbb{R}^{n\times n}$ is positive real, the following problem occurs:
By lemma 5.8 the explicit representation of the univariate decrease rates $\theta_j$ from definition 5.7 involves in the nominator the expressions

$$u_j^T A^T M r_{J_k}, \tag{5.32}$$

where $1 \le k \le n$ is such that the considered projection method in $J_k$-explicit and $j$ is an element of the corresponding candidate set $C_{J_k}$. Depending on the particular choice of the projection matrices $Z_L$ and $Z_R$, the quantity from (5.32) is possibly unknown:

$$\begin{aligned} u_j^T A^T M r_{J_k} &= \frac{1}{2} u_j^T A^T Z_L A^{-1} r_{J_k} + \frac{1}{2} u_j^T A^T A^{-T} Z_L^T r_{J_k} \\ &\overset{(4.9)}{=} \frac{1}{2}\left(u_j^T A^T Z_L A^{-1}\left(APZ_R u_k - Z_R u_k\right)\right) + \frac{1}{2} u_j^T Z_L^T r_{J_k} \\ &= \frac{1}{2}\big(\underbrace{u_j^T A^T Z_L P Z_R u_k}_{\text{known}} - \underbrace{u_j^T A^T Z_L A^{-1} Z_R u_k}_{\text{possibly unknown}}\big) + \frac{1}{2} \underbrace{u_j^T Z_L^T r_{J_k}}_{\text{known}}. \end{aligned} \tag{5.33}$$

Thus, if the quantities $u_j^T A^T Z_L A^{-1} Z_R u_k$ are unknown, the univariate decrease rates from definition 5.7 in the form stated by lemma 5.8 are unknown as well. In this case, we consider the following two estimations for the univariate decrease rates:

$$\tilde{\theta}_j^1 := \frac{\left(u_j^T A^T Z_L^T P Z_R u_k + u_j^T Z_L^T r_{J_k}\right)^2}{\left\|u_j\right\|_{Z_L^T A}^2}, \tag{5.34}$$

and

$$\tilde{\theta}_j^2 := \frac{\left(u_j^T Z_L^T r_{J_k}\right)^2}{\left\|u_j\right\|_{Z_L^T A}^2}, \tag{5.35}$$

where $1 \le j, k \le n$, the considered projection method is $J_k$-explicit, and $j \in C_{J_k}$. The estimates $\tilde{\theta}_j^1$ in (5.34) are obtained by ignoring the unknown addend in (5.33), and the estimates $\tilde{\theta}_j^2$ are obtained by substituting the matrix $A^{-1}$ in the unknown term of (5.33) by $P$. In chapter 7, we survey the numerical properties of projection methods for which the pattern adaption process is based on the estimations $\tilde{\theta}_j^1$ from (5.34) and $\tilde{\theta}_j^2$ from (5.35), and we compare those to the results obtained by utilizing the decrease rates $\lambda_j$ in the form stated by theorem 5.5.

**The Relation between the Decrease Rates $\lambda_j$ and the Univariate Decrease Rates $\theta_j$**

With the explicit representation of the univariate decrease rates $\theta_j$ given by lemma 5.8, we see that the $\theta_j$ are closely related to the decrease rates $\lambda_j$ from definition 5.1: if the product matrix $Z_L^H A$ is Hermitian positive definite, the univariate decrease rates are

$$\theta_j = \frac{|\ u_j^H Z_L^H r_{J_k}\ |^2}{\left\|u_j\right\|^2_{Z_L^H A}},$$

and the decrease rates $\lambda_j$ are by theorem 5.5

$$\lambda_j = \frac{|\ u_j^H Z_L^H r_{J_k}\ |^2}{\left\|u_j\right\|^2_{Z_L^H A} - g_j^H y_j},$$

where the vector $g_j \in I\!K^{\# J_k}$ is defined by

$$g_j := \left(Z_L^H A\right)(J_k, j),$$

and the vector $y_j \in I\!K^{\# J_k}$ is the solution of the $(\# J_k \times \# J_k)$-linear system

$$\left(Z_L^H A\right)(J_k, J_k) y_j = g_j.$$

The difference between the $\theta_j$ and the $\lambda_j$ is the presence of the dot-product $g_j^H y_j$, and the involved $(\# J_k \times \# J_k)$-linear system in the denominator of the $\lambda_j$. Thus, calculating the univariate decrease rates $\theta_j$ is computationally cheaper than the calculation of the decrease rates $\lambda_j$.
However, which of those two approaches leads to a better overall performance of the preconditioned iteration solve is not known in advance. This issue is discussed in detail in chapter 7.

The following lemma shows a basic relation between the univariate decrease rates $\theta_j$ from definition 5.7 and the corresponding decrease rates $\lambda_j$ from definition 5.1. The statement of this lemma is the generalization of the corresponding statement given in [17] for a particular projection method.

**Lemma 5.9** (Relation Between the Decrease Rates)
*Let the matrix $P$ be the projective approximate inverse of the non-singular matrix $A \in I\!K^{n\times n}$ determined by a projection method with the projection matrices $Z_L,\ Z_R \in I\!K^{n\times n}$ on the projection pattern $(J_1,\ldots,J_n)$. Let further the matrix $Z_LA^{-1} \in I\!K^{n\times n}$ be Hermitian positive definite. Then for $1 \le j,k \le n$ with $j \notin J_k$ the inequalities*

$$0 \le \theta_j \le \lambda_j \tag{5.36}$$

*and*

$$\lambda_j = 0 \Longleftrightarrow \quad \theta_j = 0 \tag{5.37}$$

*hold, where the $\lambda_j$ are the decrease rates from definition 5.1 and $\theta_j$ denotes the estimated decrease rates from definition 5.7.*

**Proof.**

We prove assertion (5.36):

$$\begin{aligned}
\lambda_j &\overset{(5.3)}{=} \left\|r_{J_k}\right\|^2_{Z_LA^{-1}} - \left\|r_{J_k\cup(j)}\right\|^2_{Z_LA^{-1}} \\
&\overset{(4.19)}{=} \left\|r_{J_k}\right\|^2_{Z_LA^{-1}} - \min_{\substack{v\in I\!K^n_{J_k}\\ \zeta\in I\!K}} \left\|Av + \zeta Au_i - Z_Ru_k\right\|^2_{Z_LA^{-1}} \\
&\ge \left\|r_{J_k}\right\|^2_{Z_LA^{-1}} - \min_{\zeta\in I\!K} \left\|r_{J_k} + \zeta Au_i\right\|^2_{Z_LA^{-1}} \\
&\overset{(5.29)}{=} \theta_j \\
&\ge 0.
\end{aligned}$$

Assertion (5.37) follows by noting that both $\lambda_j$ and $\theta_j$ are fractions with the same nominator $u_j^H Z_L^H r_{J_k}$.

$\diamondsuit$

The above lemma shows that the univariate decrease rates $\theta_j$ are closely related to the decrease rates $\lambda_j$. However, our numerical tests presented in chapter 7 show, that in practice it is not at all obvious which variety of decrease rates leads to more efficient preconditioners.

If the considered projection method is such that the matrix $Z_L^H A$ is Hermitian positive definite, lemma 5.6 applies. Hence in this case, if for some $1 \le k \le n$ the $J_k$-residual is non-vanishing, there exists at least one index $j \in C_{J_k}$, for which the corresponding univariate decrease rate $\theta_j$ is non-vanishing. Thus, in practice it suffices to determine the univariate decrease rates $\theta_j$ only for $j \in L_{J_k}$, with the set $L_{J_k}$ as defined in (5.23), i.e.

$$L_{J_k} := \left\{ j \mid \left(Z_L^H r_{J_k}\right)_j \ne 0,\ j \notin J_k \right\},$$

for $k = 1,\ldots,n$.

**The `Univariate Pattern Adaptive Projective Approximate Inverse` Algorithm**

We give a pseudo-code formulation of the `Pattern Adaptive Projective Approximate Inverse` algorithm (algorithm 8), including the univariate decrease rates in the form stated by lemma 5.8. The input data and the parameters for this algorithm are identical to those required for algorithm 8 (see pages 67–73). On output, this algorithm returns a projective approximate inverse $P$ on an adaptively derived projection pattern.

1. Compute the projective approximate inverse $P^0$ with the initial projection pattern $(J_1^0, \ldots, J_n^0)$ using algorithm 6
2. $r_{J_k^0} := (AP^0 - I)\, Z_R u_k$ for $k = 1, \ldots, n$
3. For $l = 1, \ldots, ms$
   (a) For $k = 1, \ldots, n$
      i. If $\left( \left\| r_{J_k^{l-1}} \right\|_{Z_L A^{-1}} < \epsilon_k \right)$ Cycle $k$-loop
      ii. $mt := \min\left(mfps, mf - \left(\#J_k^{l-1}\right)\right)$
      iii. If $(mt == 0)$ Cycle $k$-loop
      iv. Determine the set $L_{J_k^{l-1}}$ as defined in (5.23), and determine for all $j \in L_{J_k^{l-1}}$ the univariate decrease rates
      $$\theta_j := \frac{|\left(Z_L^H r_{J_k}\right)_j|^2}{\left\| u_j \right\|_{Z_L^H A}^2}$$
      v. Determine the $mt$ indices $j_1, \ldots, j_{mt}$ according to the largest estimated decrease rates $\theta_j$
      vi. $J_k^l := J_k^{l-1} \cup (j_1, \ldots, j_{mt})$
   (b) If no column-pattern was enlarged in step (3a) : STOP
   (c) Compute the approximate inverse $P^l$ with the projection pattern $\left(J_1^l, \ldots, J_n^l\right)$ using algorithm 6
   (d) If $(l \neq ms)$: Compute the new $J_k^l$-residuals $r_{J_k^l}$ for $k = 1, \ldots, n$

**Algorithm 11:** `Univariate Pattern Adaptive Projective Approximate Inverse`

The above algorithm, as well as the input data and the parameters, are identical with algorithm 8 except of step (3(a)iv) where the univariate decrease rates are determined.
If the matrix $Z_L A^{-1} \in I\!R^{n\times n}$ is non-symmetric or indefinite, the quantities $\tilde{\theta}_j^{1,2}$ from (5.34) or (5.35) can be used instead of the univariate decrease rates $\theta_j$ from definition 5.7.

**The Computational Cost of the Univariate Decrease Rates**
The computational cost of the above algorithm, caused by calculating the univariate decrease rates in step (3(a)iv), depends on the particular choice of the matrix $Z_L$: For the univariate decrease rates $\theta_j$ and for the estimates $\tilde{\theta}_j^2$ from (5.35), the following operatins are necessary: for determining the sets $L_{J_k^{l-1}}$, the sparse dot-products $u_j^H Z_L^H r_{J_k^{l-1}}$, for $j \notin J_k^{l-1}$, are determined. Additionally, the sparse dot-products $u_j^H Z_L^H A u_j$ for the denominator are determined. The left-hand side projection matrix $Z_L$ should be such that those dot-products are computationally cheap.
If the $\tilde{\theta}_j^1$ from equation (5.34) are applied as estimates for the univariate decrease rates from definition 5.7, for the nominator two (possibly sparse) matrix-vector multiplications, namely $PZ_R u_k$ and $A\,(PZ_R u_k)$, and one sparse dot-product, namely $u_j^T A^T\,(AP_R Z_R u_k)$, are necessary. The quantities $u_j^H Z_L^H A u_j$ for the nominator of the decrease rates are already known from determining the sets $L_{J_k^{l-1}}$.

In addition to the potential for vectorization and parallelization of the above algorithm, as discussed for algorithm 8 (see page 71), the calculation of the univariate decrease rates in step (3(a)iv) of the above algorithm yields further potential for vectorization and parallelization. The univariate decrease rates can all be determined independently from each other, and the operations for determining them are dot-products.

**Optimizations for the Practical Implementation**
The remarks given on pages 72–73 regarding possible optimizations of algorithm 8, and the remarks given for algorithm 10 on pages 87–87 apply.

**The Case of diagonal $Z_R$**
For projection methods with a diagonal right-hand side projection matrix $Z_R$, the projective approximate inverse $P$ can be determined with algorithm 9, where the calculation of the univariate decrease rates $\theta_j$ in step (1(c)iv) of algorithm 9 is done in the same way as in step (3(a)iv) of algorithm 11. With the resulting algorithm, the columns $p_k$ of the projective approximate inverse $P$, and the corresponding column patterns, can be determined independently from each other.

In this section, we have developed a new strategy for estimating the decrase rates $\lambda_j$ from definition 5.1 for general projection methods. We have discussed the computational complexity of these estimated decrease rates and we have cosidered algorithms 8 and 9 including the calculation of these estiamted decrease rates.

Altogether, in this chapter, we have developed a set of new algorithms for the calculation of projective approximate inverses on adaptively determined projection patterns for general projection methods. These algorithms differ in the strategy for the pattern derivation: algorithm 10 involves the decrease rates $\lambda_j$ from definition 5.1, and algorithm 11 involves the univariate decrease rates $\theta_j$ from definition 5.7, or estimates of these univariate decrease rates. Furthermore, we have considered the special case of projection methods with a diagonal right-hand side projection matrix $Z_R$, as well as the resulting algorithms for the adaptive pattern derivation by utilizing the decrease rates $\lambda_j$ from definition 5.1 and the univariate decrease rates $\theta_j$ from definition 5.7. Importantly, all these new algorithms are inherently parallel and highly vectorizable.
In the following chapter, we consider some state-of-the-art preconditioning techniques in greater detail. Some of those preconditioning techniques are classified in terms of projection methods. For these methods, the theoretical results from chapter 4 and the adaptive pattern derivation strategies deduced in this chapter apply. Further, we propose new preconditioning techniques, which are projection methods. Those methods are obtained by making particular choices for the projection matrices $Z_L$ and $Z_R$.
The numerical properties of some of the state-of-the-art preconditioning methods and of the new projection method-based techniques are compared in chapter 7.

# 6 Practical Preconditioning Algorithms

The design of efficient solvers for large sparse linear systems on parallel computers is an important task for today's scientific computing research. Krylov subspace methods for solving linear systems (see section 3.5) can efficiently be implemented on parallel computers, but since those methods may fail to converge, some preconditioning, prior to the application of an iterative solver for the linear system, is necessary.
During the last years, a lot of research has been devoted to preconditioning iterative methods for large sparse linear systems, and many methods have been proposed. While many of these preconditioning methods are efficient for certain types of linear systems and, also, for certain types of computer architectures, no robust, efficiently parallelizable preconditioning method is known up to now.
On the one hand, there are methods based on incomplete factorizations (like `ILU`, see section 6.4). These methods have been developed and improved for many years, and they are known for being efficient with regards to their acceleration of the iterative solve. But those methods are inherently sequential, both in the set-up phase for the construction of the preconditioner and during the iterative solve. Thus, in general those methods are not well-suited for parallel computers.
On the other hand, there are various explicit preconditioning methods with different degrees of parallelism in the set-up phase (like the recently proposed `SPAI` method (see section 6.2 and e.g. [39]) and the `AINV` method (see section 6.6 and e.g. [9])), and full parallelism during the iterative solution process. But, in general, these methods seem to be less robust and less efficient as for the acceleration of the iterative solve, as compared to implicit methods. An ample comparison of various preconditioning techniques is given e.g. in [11].

In this chapter, we briefly sketch some of the known preconditioning methods. Furthermore, we fit three preconditioning techniques, proposed by Kolotilina and Yeremin in [46] for fixed sparsity patterns, into the theoretical framework of projection methods. Thereby we not only obtain the new approximation statements given in section 4.2 for these methods, but too we obtain new preconditioning algorithms, namely the `Plain projection` algorithm (see section 6.3), the $\mathtt{L}^\mathtt{T}$`L-projection` algorithm (see section 6.7.1) and the `LU-projection` algorithm (see section 6.7.2), by combining the approaches from [46] with the adaptive pattern derivation strategies developed in chapter 5. The `Plain projection` algorithm, which can be applied to general linear systems, furnishes an approximate inverse of the coefficient matrix of the considered linear system. The $\mathtt{L}^\mathtt{T}$`L-projection` algorithm and the `LU-projection` algorithm approximate the inverses of triangular factorizations of the coefficient matrices.
Importantly, these newly proposed preconditioning algorithms offer a large amount of parallelism, and potential for vectorization, and are thus well-suited for the implementation on supercomputers.

## 6.1 Normalization in Terms of Projection Methods

A common preconditioning technique is to apply some scaling to the linear system $Ax = b$ from the left-hand side with a diagonal matrix $P \in I\!K^{n \times n}$ in order to balance the equations of a linear system. This preconditioning approach is called normalization. Numerous choices for the diagonal entries of the scaling matrix $P$ are possible. See [80], pp. 155-157, for a discussion of normalization.

Let the diagonal elements of the left-hand side scaling matrix $P$ be denoted by $p_k$ for $k = 1, \ldots, n$. Common choices for the $p_k$ are:

$$p_k := \frac{1}{a_{kk}}, \tag{6.1}$$

$$p_k := \frac{\bar{a}_{kk}}{\left\| A^H u_k \right\|_2^2}, \tag{6.2}$$

$$p_k := \frac{1}{\left\| A^H u_k \right\|_2^2}, \tag{6.3}$$

$$p_k := \frac{\operatorname{sign}(a_{kk})}{\sum_{l=1}^{n} |a_{kl}|}, \tag{6.4}$$

where $\bar{a}_{kk} \in I\!K$ denotes the complex conjugated number $a_{kk}$. The normalization approaches corresponding to (6.1), (6.2) and (6.4) are applicable only if the matrix $A$ has a zero-free diagonal. Additionally, the normalization strategy (6.4) is applicable for real matrices only.

Two of these normalization approaches can be formulated in terms of projection methods:

**Lemma 6.1** (Diagonal Scaling in Terms of Projection Methods)
*Let the matrix $A \in I\!K^{n \times n}$ have a zero-free diagonal, i.e. $a_{ii} \neq 0$ for $i = 1, \ldots, n$. We consider projection methods with the projection matrices $Z_L := I$, $Z_R$ as defined below, and with the diagonal projection pattern $(J_1, \ldots, J_n)$, i.e. the pattern is defined by $J_k := (k)$ for $k = 1, \ldots, n$.*

*i) With the projection matrix $Z_R := I$, this projection method is practical according to definition 4.3 and the obtained projective approximate inverse $P$ equals the normalization matrix defined by (6.1).*

*ii) With the projection matrix $Z_R := A^H$, this projection method is practical according to definition 4.3 and the obtained projective approximate inverse $P$ equals the normalization matrix defined by (6.2).*

**Proof.**

First, we show that both of the projection methods are practical according to definition 4.3:
for the projection method from assertion $i)$ we have

$$\left(Z_L^H Z_R u_k\right)(J_k) = (u_k)(k) = 1,$$

and for the projection method from assertion $ii)$ we have

$$\left(Z_L^H Z_R u_k\right)(J_k) = \left(A^H u_k\right)(k) = a_{kk} \neq 0,$$

and thus both projection methods are practical.
Now we consider the shape of the projection matrices:
Since the normalization matrices are applied as left-hand side preconditioners, we consider the projection method

$$\begin{aligned} & u_k^H Z_L^H \left[PA - I\right] Z_R u_k && = 0 \\ \Longleftrightarrow \quad & u_k^H Z_R^H \left[A^H P^H - I\right] Z_L u_k && = 0, \end{aligned} \tag{6.5}$$

for $k = 1, \ldots, n$.
Let the number $1 \leq k \leq n$ be arbitrary fixed. By corollary 4.5 applied to the projection method defined by (6.5) with the column-pattern $J_k = (k)$, we have

$$u_k^H P^H u_k = \frac{u_k^H Z_R^H u_k}{u_k^H Z_R^H A^H u_k} \tag{6.6}$$

$$\Longleftrightarrow \quad p_k = u_k^H P u_k = \frac{u_k^H Z_R u_k}{u_k^H A Z_R u_k}. \tag{6.7}$$

Thus, with $Z_R := I$ we have

$$p_k = u_k^H P u_k = \frac{1}{a_{kk}},$$

and with $Z_R := A^H$ we have

$$p_k = u_k^H P u_k = \frac{\bar{a}_{kk}}{\left\|A^H u_k\right\|_2^2}.$$

$\diamondsuit$

Since the normalization techniques defined by (6.1) and (6.2) are projection methods, corollary 4.10 applies and we obtain the following approximation properties:

**Corollary 6.2** (Approximation Properties for Diagonal Scaling)

*i)* *Let $\mathbb{K} = \mathbb{R}$ and let the approximate inverse $P$ of the non-singular matrix $A \in \mathbb{K}^{n\times n}$ be determined by the projection method defined in assertion i) of lemma* 6.1*. Let the $k$-th row of $P$ be denoted by $p_k$ for $k = 1, \dots, n$. Then the following approximation properties hold:*

*(a) If the matrix $A^{-T} \in \mathbb{R}^{n\times n}$ is positive real, then the inequalities*

$$\left\|A^T p_k^T - u_k\right\|_{A^{-1}} \leq \gamma \cdot \min_{\zeta\in\mathbb{R}} \left\|\zeta A^T u_k - u_k\right\|_{A^{-1}} \tag{6.8}$$

*and*

$$\left\|p_k^T - A^{-T} u_k\right\|_{A} \leq \gamma \cdot \min_{\zeta\in\mathbb{R}} \left\|\zeta u_k - A^{-T} u_k\right\|_{A} \tag{6.9}$$

*hold for $k = 1, \dots, n$, with $\gamma \in R$ defined by*

$$\gamma := \sqrt{1 + \frac{\rho^2\left(R\right)}{\mu_m^2\left(M\right)}},$$

*where $M$ denotes the symmetric part and $R$ denotes the skew-symmetric part of $A^{-T}$, $\rho\left(R\right)$ denotes the spectral radius of $R$ and $\mu_m\left(M\right)$ denotes the minimal eigenvalue of $M$.*

*(b) If the matrix $A^{-H} \in \mathbb{K}^{n\times n}$ is Hermitian positive definite, then the equalities*

$$\left\|A^H p_k^H - u_k\right\|_{A^{-1}} = \min_{\zeta\in\mathbb{K}} \left\|\zeta A^H u_k - u_k\right\|_{A^{-1}} \tag{6.10}$$

*and*

$$\left\|p_k^H - A^{-H} u_k\right\|_{A} = \min_{\zeta\in\mathbb{K}} \left\|\zeta u_k - A^{-H} u_k\right\|_{A} \tag{6.11}$$

*hold for $k = 1, \dots, n$.*

*ii)* *Let $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ and let the approximate inverse $P$ of the non-singular matrix $A \in \mathbb{K}^{n\times n}$ be determined by the projection method defined in assertion ii) of lemma* 6.1*. Let the $k$-th row of $P$ be denoted by $p_k$ for $k = 1, \dots, n$. Then the equalities*

$$\left\|A^H p_k^H - u_k\right\|_{2} = \min_{\zeta\in\mathbb{K}} \left\|\zeta A^H u_k - u_k\right\|_{2} \tag{6.12}$$

*and*

$$\left\|p_k^H - A^{-H} u_k\right\|_{AA^H} = \min_{\zeta\in\mathbb{K}} \left\|\zeta u_k - A^{-H} u_k\right\|_{AA^H}, \tag{6.13}$$

*hold for $k = 1, \dots, n$.*

**Proof.**

The assertions follow directly from corollary 4.10 applied to the corresponding projection methods.

$\diamond$

The above corollary shows that the normalizations defined by (6.1) and (6.2) yield particular one-dimensional minimizations for the rows $u_k^H PA$ of the preconditioned matrix $PA$.
Although normalization is a simple and computationally cheap preconditioning technique, it may be a helpful tool for considering linear systems (see [80], pp. 155-157). However, for efficiently preconditioning "difficult" linear systems more powerful preconditioning techniques are necessary.

## 6.2 Frobenius Norm Minimizing in Terms of Projection Methods

In this section, we consider the well known Frobenius norm minimizing preconditioning approach introduced by Benson in [2]. This preconditioning approach offers a variety (depending on parameters and implementation details) of strategies for computing one-sided approximate inverses.
In [83], pp. 23-25, Zimmermann classified the Frobenius norm based preconditioning approach in terms of projection methods. Thus, the approximation estimates, and the strategies for the adaptive pattern derivation for projection methods (given in chapters 4 and 5) apply. These results for the Frobenius norm based approach have been published before for real matrices by Cosgrove, Diaz and Griewank in [17], by Huckle and Grote in [39], and by Gould and Scott in [32]. Some properties and numerical results of a parallel implementation of the `SPAI` algorithm given in [39] are presented by Deshpande, Grote, Messmer and Sawyer in [20]. A survey on a priori matrix patterns for this approach is given by Huckle in [38].

**The Frobenius Norm Ansatz**
Let the matrix $A \in \mathbb{K}^{n\times n}$ be non-singular. The right-hand side approximate inverse $P$ based on the Frobenius norm minimizing approach is determined by minimizing the Frobenius norm expression

$$\left\|AP - I\right\|_F^2 = \sum_{k=1}^{n} \left\|(AP-I)u_k\right\|_2^2, \tag{6.14}$$

where the approximate inverse $P \in \mathbb{K}^{n\times n}$ has some prescribed matrix-pattern $S_P$, and $u_k \in \mathbb{K}^n$ denotes the $k$-th unit vector.

Let the columns of the pattern $S_P$ be denoted by $J_k$ for $k = 1, \ldots, n$. Then minimizing the Frobenius norm expression from (6.14) is equivalent to solving the $n$ small least-squares problems

$$\min_{w \in I\!K^n_{J_k}} \left\| Aw - u_k \right\|_2^2 \tag{6.15}$$

for $k = 1, \ldots, n$, where the $k$-th column $p_k \in I\!K^n_{J_k}$ of the approximate inverse $P$, is the solution of the $(\#J_k \times \#J_k)$ least-squares problems (6.15) for $k = 1, \ldots, n$. Since the $n$ small least-squares problems (6.15) are independent from each other, they can be solved in parallel.

**Classification in Terms of Projection Methods**

The Frobenius norm minimizing approach for real matrices is classified in terms of projection methods by Zimmermann in [83] (pp. 23-25). We catch on to this argumentation, extended to the complex case:

Let the matrix $A \in I\!K^{n \times n}$ be non-singular. We consider projection methods with the projection matrices $Z_L = A$ and $Z_R = I$ and with some projection pattern $(J_1, \ldots, J_n)$, i.e.

$$u_i^H A^H \left[ AP - I \right] u_k = 0 \tag{6.16}$$

for $i \in J_k$ and $k = 1, \ldots, n$. Because the product matrix $A^H A$ is Hermitian positive definite, those projection methods are explicit according to definition 4.3, and thus the corresponding projective approximate inverses $P$ are well-defined and have the unique explicit representation stated in corollary 4.5. If the vectors $\left(A^H u_k\right)(J_k)$ are non-vanishing for all $k = 1, \ldots, n$, those projection methods are practical according to definition 4.3. The candidate sets $C_{J_k}$ from definition 5.1 for this projection method are

$$C_{J_k} = \{1, \ldots, n\} \setminus J_k.$$

The approximation statement given by corollary 4.10 for those projection methods coincides with the minimization approach (6.15), and hence with the construction principle of this preconditioning technique.

Thus, all results and algorithms as for the adaptive pattern derivation for projective approximate inverses given in chapter 5 apply for the Frobenius norm minimizing approach. These pattern adaption strategies have been investigated before for this particular projection method in [17], [32] and [39]. In particular, for the adaptive pattern derivation, lemma 5.8 coincides with the results given by Huckle in [39] for $I\!K = I\!R$, and theorem 5.5 coincides with the approach proposed by Cosgrove, Diaz and Griewank in [17] for $I\!K = I\!R$.

The $J_k$-residuals from definition 4.6 for the Frobenius norm based approach have the form

$$r_{J_k} = Ap_k - u_k,$$

where the $J_k$ for $k = 1, \ldots, n$ denote some column-patterns and $p_k$ denotes the $k$-th column of the projective approximate inverse obtained from the projection method

(6.16). Since the residuals are minimized in the Euclidian norm, no Residual-Minimizing smoothing is needed for the qualitative stopping criterion of the pattern derivation process.
We give the pseudo-code of the Frobenius norm based projection method as presented in [39], which is identical to algorithm 9 with $Z_L := A$. In [39], this algorithm is called `SPAI`, in abbreviation of SParse Approximate Inverse. The input data and the parameters needed for this algorithm are the same as for algorithm 8 (see pages 67–69).

1. For $k = 1, \ldots, n$
   (a) Compute the column $p_k^0$ of the projective approximate inverse $P^0$ with the initial column-pattern $J_k^0$ by solving the least-squares problem

   $$\min_{p_k^0 \in I\!K^n_{J_k^0}} \left\| A p_k^0 - u_k \right\|_2^2$$

   (b) $r_{J_k^0} := A p_k^0 - u_k$
   (c) For $l = 1, \ldots, ms$
      i. If $\left( \left\| r_{J_k^{l-1}} \right\|_2 < \epsilon_k \right)$ Cycle $k$-loop
      ii. $mt := \min \left( mfps, mf - \left( \# J_k^{l-1} \right) \right)$
      iii. If $(mt == 0)$ Cycle $k$-loop
      iv. Determine the set $L_{J_k^{l-1}} := \left\{ j \mid \left( A^H r_{J_k^{l-1}} \right)_j \neq 0,\ j \notin J_k^{l-1} \right\}$ from (5.23)
      v. Compute the univariate decrease rates $\theta_j$ from definition 5.7 for $j \in L_{J_k^{l-1}}$
      vi. Determine the $mt$ indices $j_1, \ldots, j_{mt}$ according to the largest univariate decrease rates $\theta_j$
      vii. $J_k^l := J_k^{l-1} \cup (j_1, \ldots, j_{mt})$
      viii. Compute the $k$-th column $p_k^l$ of $P^l$ on the column-pattern $J_k^l$ by solving the least-squares problem

      $$\min_{p_k^l \in I\!K^n_{J_k^l}} \left\| A p_k^l - u_k \right\|_2^2$$

      ix. If $(l \neq ms)$: Compute the new $J_k^l$-residual $r_{J_k^l}$

**Algorithm 12:** `SPAI`

For a discussion of the algortihmic flow and of the computational complexity of the `SPAI` algorithm, the general remarks for algorithm 9 apply with $Z_L := A$ (see pages 73–75).
In particular, the steps that contribute to the computational complexity of the `SPAI` algorithm are:

**steps** (1a), (1(c)viii) In these steps, the least-squares problems

$$\min_{p_k^i \in \mathbb{K}^n_{J_k^i}} \left\| A p_k^i - u_k \right\|_2^2$$

with $0 \leq i \leq ms$ have to be solved at most $ms + 1$ times for each column of the approximate inverse, i.e. for $k = 1, \ldots, n$. An optimized approach for solving these least squares problems is given in [39].

**steps** (1b), (1(c)ix) Here, the new $J_k^l$-residuals are computed requiring at most the $ms + 1$ sparse matrix-vector products $Ap_k^l$ for $k = 1, \ldots, n$.

**step** (1(c)i) In these steps, the Euclidian norms of $J_k^l$-residuals $r_{J_k^l}$ are computed at most $ms$ times for $k = 1, \ldots, n$.

**step** (1(c)iv) For determining the sets $L_{J_k^{l-1}}$, the sparse dot-products $u_j^H A^H r_{J_k^{l-1}}$ are calculated for $j \notin J_k^{l-1}$. This step of the algorithm may be a problem both in a sequential and in a parallel environment.
On a sequential computer, this step is decicive for the number of floating-point operations, and thus for the overall construction time of the approximate inverse. Some heuristic strategies for restricting the number of indices $j$ are given in [39].
On parallel computers, the amount of communication for transferring the columns $Au_j$ of the matrix $A$ between the processors may be prohibitively high, especially if the dot-products $u_j^H A^H r_{J_k}$ are formed for all indices $j \notin J_k^{l-1}$. This problem may be alleviated by restricting the number of dot-products that actually are formed. Such an approach for a parallel implementation is considered in [20]. Theoretically, the communication overhead caused by forming the dot-product $u_j^H A^H r_{J_k}$ can also be circumvented by forming the sparse matrix-vector product $A^H r_{J_k}$ first (this operation can efficiently be done in parallel), and then extracting the $j$-th component of the resulting vector. But such a approach is impractical, since this involves a prohibitively large number of floating-point operations.

**step** (1(c)v) For the pattern derivation of the `SPAI` algorithm, both the decrease rates $\lambda_j$ from definition 5.1 and the univariate decrease rates $\theta_j$ from definition

5.7 can be utilized:
By lemma 5.8, the univariate decrease rates $\theta_j$ have the form

$$\theta_j = \frac{| u_j^H A^H r_{J_k} |^2}{\left\| A u_j \right\|_2^2}. \tag{6.17}$$

Thus, the quantity $\left\| A u_j \right\|_2^2$ must be computed for each univariate decrease rate $\theta_j$ . The quantities $u_j^H A^H r_{J_k}$ are already known from step (1(c)iv).
By theorem 5.5, the decrease rates $\lambda_j$ from definition 5.3 for the multivariate approach have the form

$$\lambda_j = \frac{| u_j^H A^H r_{J_k} |^2}{\left\| A u_j \right\|_2^2 - g_j^H y_j}, \tag{6.18}$$

with the vector $g_j \in I\!K^{\# J_k}$ defined by

$$g_j := \left(A^H A\right)(J_k, j),$$

and with the vector $y_j \in I\!K^{\# J_k}$ being the solution of the $(\# J_k \times \# J_k)$-linear system

$$\left(A^H A\right)(J_k, J_k) y_j = g_j.$$

Thus, the vector $y_j$ additionally must be determined and the dot-product $g_j^H y_j$ must be evaluated, at least in theory, for the decrease rates $\lambda_j$. An optimized approach for calculating the decrease rates $\lambda_j$ is proposed in [32].

We present numerical results of the `SPAI` algorithm in chapter 7.

## 6.3 The `Plain projection` method

In this section, we consider a preconditioning approach introduced by Kolotilina and Yeremin in [46] for a fixed matrix-pattern of the preconditioner. This approach was classified by Zimmermann in [83] in terms of projection methods:

$$u_i^H [AP - I] u_k = 0 \tag{6.19}$$

for some fixed projection pattern $(J_1, \ldots, J_n)$ with $i \in J_k$ and $k = 1, \ldots, n$, i.e. the algorithm introduced in this section is obtained by making the choice $Z_L := Z_R := I$ for the projection matrices $Z_L$ and $Z_R$ from definition 4.2.
We catch on to the projection method formulation (6.19) of this approach and we combine this projection method with the adaptive pattern derivation strategies

developed in chapter 5. We designate the resulting preconditioning technique (6.19) by "`Plain projection`" method.

Theoretically, the `Plain projection` method can be utilized for calculating approximate inverses of both Hermitian and non-Hermitian matrices. But, if the considered matrix is Hermitian, the preconditioned linear system will not necessarily be Hermitian. Thus, for Hermitian matrices, other preconditioning techniques, e.g. the `L`$^{\mathtt{T}}$`L-projection` method (deduced in section 6.7.1), should be applied. For completeness, in the following we consider both Hermitian, and non-Hermitian matrices.

**Theoretical Properties**

By applying corollary 4.5 to the `Plain projection` method (as stated in (6.19)), we obtain formulae for the explicit calculation of the columns $p_k$ of the projective approximate inverse $P$:

**Corollary 6.3** (Explicit Representation for the `Plain projection` method)
*Let the matrix $A \in I\!K^{n\times n}$ be non-singular. We consider the* `Plain projection` *method from* (6.19) *with a projection pattern* $(J_1, \ldots, J_n)$. *If the* `Plain projection` *method is $J_k$-explicit according to definition* 4.3 *for some $k$, then the $k$-th column $p_k$ of the projective approximate inverse $P$ has the unique explicit representation*

$$(p_k)_j = \begin{cases} \left( [A(J_k, J_k)]^{-1} (u_k)(J_k) \right)_i & \text{for } j = (J_k)_i, \\ 0 & \text{else.} \end{cases} \tag{6.20}$$

*If the relation $k \in J_k$ holds for $k = 1, \ldots, n$ and the* `Plain projection` *method is explicit, then the* `Plain projection` *method is practical according to definition* 4.3.

**Proof.**

The assertion follows directly from corollary 4.5 applied to the `Plain projection` method (6.19).

◇

Thus, the $J_k$-reduced $k$-th column $(p_k)(J_k)$ of the projective approximate inverse $P$ determined with the `Plain projection` method equals the $l$-th column of the matrix

$$[A(J_k, J_k)]^{-1},$$

where $l$ is such that $(J_k)_l = k$.

The following corollary states the approximation properties of the projective approximate inverse determined with the `Plain projection` method to the exact inverse of $A$.

**Corollary 6.4** (Approximation Properties for the `Plain projection` method)
*We consider the* `Plain projection` *method from* (6.19) *with a projection pattern* $(J_1, \ldots, J_n)$ *for a non-singular matrix* $A \in I\!K^{n\times n}$. *Let* $p_k$ *denote the* $k$*-th column of the projective approximate inverse* $P$ *for* $k = 1, \ldots, n$.

*i) Let the* `Plain projection` *method be* $J_k$*-explicit according to definition* 4.3 *for some* $k \in I\!N$ *with* $1 \le k \le n$. *Let the matrix* $A^{-1} \in I\!R^{n\times n}$ *be positive real and let* $A^{-1} = M + R$, *where* $M$ *denotes the symmetric and* $R$ *is the skew-symmetric part of* $A^{-1}$. *Let* $\rho(R)$ *denote the spectral radius of* $R$ *and let* $\mu_m(M)$ *denote the minimal eigenvalue of* $M$. *Then the estimations*

$$\left\|Ap_k - u_k\right\|_{A^{-1}} \le \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in I\!R^n_{J_k}} \left\|Aw - u_k\right\|_{A^{-1}} \tag{6.21}$$

*and*

$$\left\|p_k - A^{-1}u_k\right\|_{A^T} \le \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in I\!R^n_{J_k}} \left\|w - A^{-1}u_k\right\|_{A^T} \tag{6.22}$$

*hold.*

*ii) If* $A^{-1} \in I\!K^{n\times n}$ *is Hermitian positive definite, the equalities*

$$\left\|Ap_k - u_k\right\|_{A^{-1}} = \min_{w \in I\!K^n_{J_k}} \left\|Aw - u_k\right\|_{A^{-1}} \tag{6.23}$$

*and*

$$\left\|p_k - A^{-1}u_k\right\|_{A^H} = \min_{w \in I\!K^n_{J_k}} \left\|w - A^{-1}u_k\right\|_{A^H} \tag{6.24}$$

*hold for* $k = 1, \ldots, n$.

**Proof.**

The assertions follow directly from corollary 4.10 applied to the `Plain projection` method from (6.19).

◇

**Adaptive Pattern Derivation**

The $J_k$-residuals and the $J_k$-errors from definition 4.6 for the `Plain projection` method from (6.19) have the form

$$r_{J_k} = Ap_k - u_k$$

and

$$e_{J_k} = p_k - A^{-1}u_k$$

for $k = 1, \ldots, n$, where $p_k$ denotes the $k$-th column of the projective approximate inverse $P$.

By corollary 6.4, the $J_k$-residuals $r_{J_k}$ of the `Plain projection` method are minimized in the norm $||.||_{A^{-1}}$. Since in practical applications the matrix $A^{-1}$ is unknown, the value of $||r_{J_k}||_{A^{-1}}$ cannot be calculated. In this situation, for the qualitative stopping criterion of the pattern derivation (see pages 67–68), some other norm of the $J_k$-residual, e.g. the Euclidian norm, may be used. Alternatively, since the Euclidian norm of the $J_k$-residual does not decrease monotonously for an augmented pattern $J_k$, the Euclidian norm of the related Residual-Minimizing smoothed residuals $s_{J_k}$ (see pages 67–68 and section 3.4) can be utilized for the qualitative control of the pattern derivation. The Euclidian norm of the smoothed $J_k$-residual decreases monotonously in the Euclidian norm for growing pattern $J_k$.

The following corollary shows the shape of the univariate decrease rates $\theta_j$ from definition 5.7 and of the decrease rates $\lambda_j$ from definition 5.1 for the `Plain projection` method from (6.19):

**Corollary 6.5** (Decrease Rates for the `Plain projection` method)
*We consider the* `Plain projection` *method from* (6.19) *with a projection pattern* $(J_1, \ldots, J_n)$ *for a non-singular matrix* $A \in \mathbb{K}^{n\times n}$. *Let* $p_k$ *denote the* $k$*-th column of the projective approximate inverse* $P$ *for* $k = 1, \ldots, n$.

i) *Let the* `Plain projection` *method be* $J_k$*-explicit according to definition* 4.3 *for some* $k \in \mathbb{N}$ *with* $1 \leq k \leq n$. *Let the matrix* $A^{-1} \in \mathbb{R}^{n\times n}$ *be positive real and let* $M$ *denote the symmetric part of* $A^{-1}$, *then the univariate decrease rates* $\theta_j$ *from definition* 5.7 *have the form*

$$\theta_j = \frac{\left(u_j^T A^T M r_{J_k}\right)^2}{\left\|u_j\right\|_A^2} \tag{6.25}$$

*for all elements* $j$ *of the candidate set* $C_{J_k}$ *from definition* 5.1.

*ii) Let the matrix $A^{-1} \in I\!\!K^{n\times n}$ be Hermitian positive definite, then the univariate decrease rates $\theta_j$ from definition 5.7 have the form*

$$\theta_j = \frac{|\, u_j^H r_{J_k} \,|^2}{\left\|u_j\right\|_A^2} \tag{6.26}$$

*for $1 \le j \le n$ with $j \notin J_k$ and for $k = 1, \dots, n$.*

*iii) Let the matrix $A^{-1} \in I\!\!K^{n\times n}$ be Hermitian positive definite. Then the decrease rates $\lambda_j$ from definition 5.1 can be written as*

$$\lambda_j = \frac{|\, u_j^H r_{J_k} \,|^2}{\left\|u_j\right\|_A^2 - g_j^H y_j}, \tag{6.27}$$

*where the vector $g_j \in I\!\!K^{\#J_k}$ is defined by*

$$g_j := A(J_k, j), \tag{6.28}$$

*and the vector $y \in I\!\!K^{\#J_k}$ is the solution of the $(\#J_k \times \#J_k)$-linear system*

$$A(J_k, J_k) y_j = g_j. \tag{6.29}$$

**Proof.**

The assertions follow directly from lemma 5.8 and theorem 5.5 applied to the `Plain projection` method from (6.19).

$\diamondsuit$

**The Univariate Decrease Rates in the Positive Real Case**

For Hermitian positive definite $A^{-1} \in I\!\!K^{n\times n}$, the univariate decrease rates $\theta_j$ as represented in (6.26) or the decrease rates $\lambda_j$ as stated in (6.27) can be utilized for the pattern derivation process of the `Plain projection` method.

If the matrix $A$ is positive real, the nominator of the univariate decrease rates $\theta_j$ as represented in (6.25) is unknown in practical applications (see the discussion on pages 92–93). In this situation, we have for the quantity in the brackets of the nominator of (6.25):

$$u_j^T A^T M r_{J_k} \stackrel{(5.33)}{=} \frac{1}{2} \underbrace{\left(u_j^T A^T P u_k\right.}_{\text{known}} - \underbrace{\left.u_j^T A^T A^{-1} u_k\right)}_{\text{unknown}} + \frac{1}{2} \underbrace{\left(u_j^T r_{J_k}\right)}_{\text{known}}.$$

In this situation, the estimates $\tilde{\theta}_j^1$ and $\tilde{\theta}_j^2$ for the univariate decrease rates $\theta_j$ as proposed in (5.34) and (5.35), which are

$$\tilde{\theta}_j^1 := \frac{\left(u_j^T A^T P u_k + u_j^T r_{J_k}\right)^2}{\left\|u_j\right\|_A^2} \tag{6.30}$$

and

$$\tilde{\theta}_j^2 := \frac{\left(u_j^T r_{J_k}\right)^2}{\left\|u_j\right\|_A^2} \tag{6.31}$$

for the `Plain projection` method, can be applied for the pattern derivation process.

**The `Plain projection` Algorithm**
In algorithm 13, we give a pseudo-code formulation of the `Plain projection` algorithm. The input data and the parameters needed for the `Plain projection` algorithm are the same as for algorithm 8 (see pages 67–69). Note that the initial projection pattern $(J_1^0, \ldots, J_n^0)$ should be such that $k \in J_k$ for all $k = 1, \ldots, n$, since only in this situation the `Plain projection` method can be practical according to definition 4.3. On output of the `Plain projection` algorithm, we obtain a projective approximate inverse determined with the `Plain projection` method on an adaptively determined projection pattern.
For an explanation of the single steps and the algorithmic flow of the `Plain projection` algorithm, we refer you the comments given to algorithm 9 on pages 73–75.

In step (1(c)v) of the `Plain projection` algorithm (as stated in algorithm 13), the univariate decrease $\theta_j$ rates from (6.26) are applied for the pattern derivation. Alternatively, the estimates $\tilde{\theta}_i^1$ from (6.30) or $\tilde{\theta}_i^2$ from (6.31) of the univariate decrease rates, or the decrease rates $\lambda_i$ from (6.27) may be applied.

The steps that contribute to the computational complexity of the `Plain projection` algorithm are:

**steps 1a,1(c)viii** In these steps, the $k$-th column $p_k^l$ of the projective approximate inverse $P^l$ on the pattern $J_k^l$ is computed by solving a linear system with the $\left(\#J_k^l \times \#J_k^l\right)$-coefficient matrix $A\left(J_k^l, J_k^l\right)$ for $k = 1, \ldots, n$.

**steps 1b, 1(c)ix** For the computation of the $J_k^l$-residuals, the sparse matrix-vector product $Ap_k^l$ is formed for $k = 1, \ldots, n$. For the calculation of the Residual-Minimizing smoothed sequence $s_{J_k^l}$ in step 1(c)ix, two dot-products, one vector-sum and one triadic operation are necessary (see algorithm 1) for $k = 1, \ldots, n$. Depending on the number of non-zeros in the corresponding $J_k^l$-residual, these operations are possibly sparse.

1. For $k = 1, \ldots, n$
   (a) Compute the $k$-th column $p_k^0$ of the projective approximate inverse $P^0$ on the column-pattern $J_k^0$ by solving the linear system $A\left(J_k^0, J_k^0\right) p_k^0 = u_k\left(J_k^0\right)$
   (b) Compute the $J_k^0$-residual $r_{J_k^0} := Ap_k^0 - u_k,\ s_{J_k^0} := r_{J_k^0}$
   (c) For $l = 1, \ldots, ms$
      i. If $\left(\left\|s_{J_k^{l-1}}\right\|_2 < \epsilon_k\right)$ Cycle $k$-loop
      ii. $mt := \min\left(mfps, mf - \left(\#J_k^{l-1}\right)\right)$
      iii. If $(mt == 0)$ Cycle $k$-loop
      iv. Determine the set $L_{J_k^{l-1}}$ defined by
      $$L_{J_k^{l-1}} := \left\{j \notin J_k^{l-1} \mid \left(r_{J_k^{l-1}}\right)_j \neq 0\right\}$$
      v. Determine the univariate decrease rates $\theta_j$ from (6.26) for all $j \in L_{J_k^{l-1}}$
      vi. Determine the $mt$ indices $i_j, \ldots, j_{mt}$ according to the largest decrease rates $\theta_j$
      vii. $J_k^l := J_k^{l-1} \cup (j_1, \ldots, j_{mt})$
      viii. Compute the $k$-th column $p_k^l$ of the plain projective approximate inverse $P^l$ on the column-pattern $J_k^l$ by solving the linear system $A\left(J_k^l, J_k^l\right) p_k^l = u_k\left(J_k^l\right)$
      ix. If $(l \neq ms)$ : Compute the new $r_{J_k^l}$-residual $r_{J_k^l} := Ap_k^l - u_k$, and the corresponding Residual-Minimizing smoothed $r_{J_k^l}$-residual $s_{J_k^l}$

**Algorithm 13:** `Plain projection`

**step 1(c)i** The calculation of the Euclidian norm of the Residual-Minimizing smoothed residuals $s_{J_k^l}$ involves one (possibly sparse) dot-product.

**step 1(c)iv** Determining the sets $L_{J_k^l}$ for the univariate decrease rates $\theta_j$, for the estimated univariate decrease rates $\tilde{\theta}_j^2$ from (6.31), and for the decrease rates $\lambda_j$ from (6.27) is cost-free.
For the estimated univariate decrease rates $\tilde{\theta}_j^1$ from (6.30), the sparse dot-product $u_j^T A^T p_k^l$ must be evaluated. Note that utilizing the estimates $\tilde{\theta}_j^1$ may cause problems on parallel computers with distributed memory: evaluating

the dot-product

$$u_j^T A^T p_k^l$$

for each $j \notin J_k^l$ may require a prohibitively high amount of interprocessor communication. This problem may be alleviated by considering only a subset (given by some heuristic) of indices $j \notin J_k^l$ instead of all of them. Theoretically, the communication overhead can also be circumvented by forming the sparse matrix-vector product $A^T p_k^l$ first (this operation can efficiently be done in parallel), and then extracting the desired components of the resulting vector. But such a strategy is impractical, since it involves a prohibitively large number of floating-point operations.

**step 1(c)v** In this step, the univariate decrease rates $\theta_j$ or their estimates, $\tilde{\theta}_j^1$, $\tilde{\theta}_j^2$, or the decrease rates $\lambda_j$ are calculated for each $j \in L_{J_k^l}$. Since the nominators of the decrease rates are known from determining the sets $L_{J_k^l}$ in step (1(c)iv), forming the univariate decrease rates $\theta_j$ or their estimates $\tilde{\theta}_j^1$ and $\tilde{\theta}_j^2$ requires only one floating-point operation, the division by $u_j^T A u_j$, for each $j \in L_{J_k^l}$.
If the decrease rates $\lambda_j$ from (6.27) are used, a linear system with the coefficient matrix $A\left(J_k^l, J_k^l\right)$ must be solved, and a sparse dot-product must be formed for each $j \in L_{J_k^l}$ for the nominator. Since one of the vectors in this sparse dot-product is $A\left(J_k^l, j\right)$ for each $j \in L_{J_k^l}$, this may require prohibitively much interprocessor communicating on parallel computers with distributed memory.

**Considerations for the Practical Implementation of the `Plain projection` Algorithm**

For the practical implementation of the `Plain projection` algorithm, the general remarks to algorithms 8 on pages 70–73 apply with $Z_L := Z_R := I$. For the calculation of the decrease rates, the remarks to algorithms 10 (see pages 85–87) and 11 (see pages 96–96) apply with $Z_L := Z_R := I$.

If the matrix $A$ is Hermitian positive definite, the `Plain projection` method is always explicit according to definition 4.3, and lemma 5.6 guarantees that none of the sets $L_{J_k^l}$ for $l = 1, \ldots, ms$ and $k = 1, \ldots, n$ is empty unless the corresponding $J_k^l$-residual vanishes.

If the matrix $A$ is non-symmetric or indefinite, the situation is more complicated: In general, it is not at all obvious whether or not the `Plain projection` method is explicit according to definition 4.3 for a given projection pattern $(J_1, \ldots J_n)$. If the initial pattern is diagonal and the diagonal of $A$ is zero-free, the `Plain projection` method is practical according to definition 4.3. Thus the diagonal pattern should be used as the initial pattern for the `Plain projection` algorithm. But in the course

of the pattern derivation for the `Plain projection` algorithm, the candidate sets $C_{J_k}$ from definition 5.1 maybe unknown. Our implementation of the `Plain projection` algorithm deals with this problem by determining the decrease rates not only for the candidate sets $C_{J_k}$ but for all $j \notin J_k$. Theoretically, our implementation can run into a singular subsystem and thus breakdown.
A further problem is that, if the corresponding $J_k$-residual is non-vanishing, the existence of a candidate index $j$ yielding a positive decreasse rate is not guaranteed by theory. Thus, theoretically the pattern derivation process possibly can stagnate for some time.
Despite of this theoretical scruples, during our extensive numerical tests none of the above problems ever occured. Conversely, as the numerical results presented in chapter 7 indicate, the `Plain projection` algorithm behaved in a benign way.

## 6.4 Incomplete $LU$-decomposition

This family of implicit preconditioning methods for non-symmetric linear systems is based on an incomplete $LU$-decomposition (`ILU`) of the matrix $A$. `ILU` methods have been investigated for decades, and many variants have been proposed. Detailed surveys on the theoretical and practical aspects of `ILU` methods are given e.g. in [40], [49] and [82]. The performance of $LU$-decompositions on vector computers is discussed in [28].
Let $S$ denote a pattern of a matrix, and let the entries of the matrix $L$ and $U$ be denoted by $l_{ij}$ and $u_{ij}$. Then the incomplete $LU$-decomposition of $A$ with the pattern $S$ has the form

$$A = LU + E \tag{6.32}$$

with a lower triangular matrix $L$ and an upper triangular matrix $U$. The entries $e_{ij}$ of the matrix $E$ are given by

$$e_{ij} = \begin{cases} 0 & \text{for } (i,j) \in S \\ a_{ij} - \sum_{l=1}^{\min(i,j)} l_{il} u_{lj} & \text{for } (i,j) \notin S \ . \end{cases} \tag{6.33}$$

The pattern $S$ can either be prescribed in advance or be derived adaptively along the computation of $L$ and $U$. If the matrix $A$ has a special structure (like matrices generated by discretization with the finite element or finite difference method), the pattern $S$ could be set equal to the pattern of $A$. For unstructured matrices the adaptive derivation of the pattern $S$ is better suited. For preconditioning purposes, the error-matrix $E$ is neglected and the product $LU$ is viewed as an approximation to $A$. Preconditioning can then be performed with

$$P_L = U^{-1} L^{-1}, \ P_R = I$$

from the left-hand side, with

$$P_L = I,\ P_R = U^{-1}L^{-1}$$

from the right-hand side, or with

$$P_L = L^{-1},\ P_R = U^{-1}$$

from both sides. The inverses $L^{-1}$ and $U^{-1}$ are not computed explicitly, since that would be computationally expensive; whenever the evaluation of a matrix-vector product $L^{-1}v$ or $U^{-1}v$ is needed, a triangular system is solved instead. Thus, two triangular systems have to be solved per iteration step.
Computing the matrices $L$ and $U$ and evaluating the triangular systems is recursive, and thus a severe bottleneck for implementation of this approach on vector and parallel computers.
On the other hand, some of today's most efficient and most robust preconditioning methods for non-symmetric linear systems, at least with regards to the acceleration of the iterative solver, belong to this family of preconditioning methods.
Many different implementations of this preconditioning approach are possible; among these are `ILUT` ([60]), `MILU` ([34]), `RILU` ([1]). Extensive overviews on `ILU`-preconditioning are given in [49], [62]. In [7], it is shown that certain reorderings of the considered linear systems may enhance the efficiency of `ILU`-preconditioners.
For our numerical tests presented in chapter 7, we use the `ILU(0)`-method as a representant for this branch of implicit preconditioning methods. This variant is characterized by allowing fill-in in the triangular factors $L$ and $U$ from (6.32) only in the locations corresponding to non-zeros in the original matrix $A$. Although this is a somewhat simple approach, the resulting preconditioners are quite efficient (see chapter 7 for details).

## 6.5 Incomplete Cholesky-decomposition

Preconditioning methods based on an incomplete Cholesky-decomposition (`IC`) of the coefficient matrix $A$ form a family of implicit methods for preconditioning symmetric positive definite linear systems. This preconditioning approach was introduced by Buleev and published in [48]. The `IC` approach was the basis for the development of the `ILU` approach. Surveys on the algorithmic properties of `IC` methods are given e.g. in [52], [76] and [77]. In [27], the performance of `IC` methods on vector computers is discussed.
`IC` methods are based on the incomplete Cholesky-decomposition

$$A = L^T L + E$$

with $L$ upper triangular. The entries $e_{ij}$ of the error-matrix $E$ are given by

$$e_{ij} = \begin{cases} 0 & \text{for } (i,j) \in S, \\ a_{ij} - \sum_{l=1}^{\min(i,j)} l_{li} l_{lj} & \text{for } (i,j) \notin S, \end{cases}$$

where $S$ is a pattern and $l_{ij}$, $a_{ij}$ denote the entries of the matrices $L$ and $A$. For the `IC`-approach, two-sided preconditioning with

$$P_L = L^{-1} \text{ and } P_R = L^{-T} \tag{6.34}$$

is used, because then the preconditioned system is symmetric.
An ample survey on the numerical properties of some variants of `IC`-methods is found in [11].
Similar to the `ILU`-approach, two triangular systems for the evaluation of $L^{-T}v$ and $L^{-1}v$ have to be solved in each iteration step. Thus, the `IC`-approach suffers from similar sequential bottlenecks as the `ILU`-approach.

For our numerical tests with symmetric coefficient matrices, we chose the `IC(0)`-method as a representant for the family of `IC`-methods. This approach is defined by calculating the triangular matrix $L$ from (6.5) which may have non-zeros only in the locations corresponding to non-zeros in the original matrix $A$. See chapter 7 for some numerical results of the `IC(0)`-method.

## 6.6 Incomplete $A$-Biconjugation

In this section, we describe a preconditioning technique introduced by Benzi, Meyer and Tůma in [6] for symmetric positive definite linear systems, and by Benzi and Tůma in [9] for non-symmetric linear systems. With this preconditioning technique the inverses of non-singular matrices are approximated with a product of two triangular matrices. Extensive surveys on this preconditioning technique are found in [4], [11] and [10]. Recent results of a parallel implementation of this preconditioning method are presented in [5].

First, we have to give a definition:

**Definition 6.6** ($A$-biconjugate vectors)
*Let $A$ be a non-singular $(n \times n)$-matrix. Then two sets of vectors $\{z_1, \ldots, z_k\}$ and $\{w_1, \ldots, w_k\}$, where $z_i$, $w_i \in I\!K^n$ for $i = 1, \ldots, n$, are called $A$-**biconjugate**, if*

$$w_i^T A z_j = 0 \iff i \neq j \tag{6.35}$$

*for $i, j = 1, \ldots, n$.*

The following considerations show the relation between the inverse of a matrix $A$ and two sets of $A$-biconjugate vectors: Let $z_{set} = \{z_1, \ldots, z_n\}$ and $w_{set} = \{w_1, \ldots, w_n\}$ be two sets of A-biconjugate vectors. Let $W = [w_1, \ldots, w_n]$ and $Z = [z_1, \ldots, z_n]$ be the matrices whose columns are the elements of the sets $w_{set}$ and $z_{set}$. Then the following equation holds:

$$W^T AZ = D = \begin{pmatrix} w_1^T Az_1 & 0 & \ldots & 0 \\ 0 & w_2^T Az_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w_n^T Az_n \end{pmatrix} \tag{6.36}$$

with $w_i^T Az_i \neq 0$ for $i = 1, \ldots, n$. Thus, the matrices $W$ and $Z$ are non-singular. Equation (6.36) is equivalent to

$$A^{-1} = ZD^{-1}W^T. \tag{6.37}$$

The matrices $W$ and $Z$ can be computed explicitly by applying a biconjugation process to the columns of two non-singular matrices $W_0$, $Z_0 \in I\!R^{n,n}$. In [9] it is suggested to set $W_0 = Z_0 = I$, because then the matrices $W$ and $Z$ will be upper triangular with ones on their diagonal. If the matrix $A$ is symmetric, only one of the triangular matrices needs to be computed, since the triangular matrices $Z$ and $W$ are identical in this case, i.e. $Z = W$.
In algorithm 14, we give the preconditioning algorithm resulting from this biconjugation approach as proposed in [9]. This algorithm is called `AINV`. In the formulation of the `AINV` algorithm (as stated in algorithm 14), $a_i^T$ denotes the $i$-th row of $A$ and $c_i^T$ denotes the $i$-th row of $A^T$. Further, $d_i$, for $i = 1, \ldots n$, denotes the diagonal entries of the matrix $D$, and $z_i$ and $w_i$ denote the columns of the matrices $Z$ and $W$ for $i = 1, \ldots, n$.

The `AINV` algorithm can be viewed geometrically as a generalized Gram-Schmidt orthogonalization process with oblique projections and with the inner product $x^T Ay$ for $x, y \in I\!R^n$.

Let $A = LMU$ be a factorization of $A$, where the matrix $L$ is lower triangular with ones on its diagonal, the matrix $M$ is diagonal (the entries of $M$ are the pivots of the corresponding $LU$-decomposition), and the matrix $U$ is upper triangular with ones on its diagonal. In exact arithmetic and without numerical dropping, the following equations for the matrices $W$, $Z$ and $D$ computed with the `AINV` algorithm hold:

$$W = L^{-T},\ Z = U^{-1},\ D = M. \tag{6.38}$$

Since the matrices $L$ and $U$ tend to be dense triangular matrices in general, in steps (2(c)iii) and (2(c)iv) of the `AINV` algorithm (algorithm 14) numerical dropping is applied to $z_j^{(i)}$ and $w_j^{(i)}$ to force the sparsity of $W$ and $Z$.

1. Set $w_i^{(0)} := z_i^{(0)} := u_i$ for $i = 1, \dots n$
2. For $i = 1, \dots, n$
   (a) For $j = i, \dots, n$
      i. $d_j^{(i-1)} := a_i^T z_j^{(i-1)}$
      ii. $q_j^{(i-1)} := c_i^T w_j^{(i-1)}$
   (b) If($i == n$) Goto 3
   (c) For $j = i+1, \dots, n$
      i. $z_j^{(i)} := z_j^{(i-1)} - \left(\frac{d_j^{(i-1)}}{d_i^{(i-1)}}\right) z_i^{(i-1)}$
      ii. $w_j^{(i)} := w_j^{(i-1)} - \left(\frac{q_j^{(i-1)}}{q_i^{(i-1)}}\right) w_i^{(i-1)}$
      iii. Apply numerical dropping to $z_j^{(i)}$
      iv. Apply numerical dropping to $w_j^{(i)}$
3. $z_i := z_i^{(i-1)}$, $w_i := w_i^{(i-1)}$, $d_i := d_i^{(i-1)}$ for $i = 1, \dots, n$

**Algorithm 14:** `AINV`

For the numerical dropping, two positive thresholds $\tau_1, \tau_2 \in I\!R$ have to be prescribed. In each step of the $i$-loop (step (2) in the `AINV` algorithm) all entries in the vectors $z_j^{(i)}$ and $w_j^{(i)}$ with an absolute value less than $\tau_1$ or $\tau_2$, respectively, are set to zero, i. e.

$$\left|\left(z_j^{(i)}\right)_l\right| < \tau_1 \Longrightarrow \left(z_j^{(i)}\right)_l := 0$$
$$\left|\left(w_j^{(i)}\right)_l\right| < \tau_2 \Longrightarrow \left(w_j^{(i)}\right)_l := 0$$

for $l = 1, \dots, n$. The numerical dropping in the `AINV` algorithm brings about a unforeseeable loss of biorthogonality: apart from changing the particular vectors to which the numerical dropping is applied, all following updates in steps (2(c)iii) and (2(c)iv) involve falsified coefficients calculated in steps (2(a)i) and (2(a)ii).

In [9], it is shown in that in exact arithmetic the `AINV` algorithm will not break down, if and only if all leading principal minors of the matrix $A$ are non-vanishing. But, due to roundoff errors in practice, a "division by zero" breakdown can occur, even if all leading principal minors of the matrix $A$ are different from zero. In [11], Benzi and Tůma suggest to simply shift "small" elements $d_i$ of the diagonal matrix $D$ to $10^{-1}$. This modification prevents the `AINV` algorithm from breakdowns, but numerical experiences with this approach indicate, that the efficiency of the obtained preconditioner decreases if more than a few of such shifts are made. Recent results of Benzi, Cullum and Tůma given in [3] and of Kharchenko, Kolotilina, Nikishin and Yeremin given in [45] consider a different strategy for preventing breakdowns of the `AINV` algorithm caused by "small" pivots. The strategy considered in both publications rests on a more elaborate way of determining the quantities $d_j^{(i-1)}$ in step (2(a)i) of the `AINV` algorithm. The numerical results presented in [3] and [45] indicate, that this modification of the `AINV` algorithm is superior to the original approach.

The computational complexity of the `AINV` algorithm depends on the chosen dropping thresholds $\tau_1$ and $\tau_2$. The larger the thresholds $\tau_1$ and $\tau_2$, and hence (at least in tendency) the sparser the vectors $z_j^{(i)}$ and $w_j^{(i)}$ are, the cheaper are the dot-products in steps (2(a)i) and (2(a)ii). In chapter 7, we give numerical results for the `AINV` algorithm.

The preconditioned system can be formed with

$$P_L = ZD^{-1}W^T, \; P_R = I$$

for left-hand side preconditioning, with

$$P_L = I, \; P_R = ZD^{-1}W^T$$

for right-hand side preconditioning, and with

$$P_L = W^T, \; P_R = ZD^{-1}$$

for two-sided preconditioning.
The matrices $W$ and $Z$ can be computed independently, and thus in parallel. If $A$ is symmetric, only the matrix $Z$ needs to be computed, since in this case $W$ must be equal to $Z$.
Since the columns of $W$ and $Z$ must be computed sequentially one after the other, an efficient implementation of the `AINV` algorithm on parallel architectures, especially with distributed memory, is at least non-trivial. Nevertheless, the results of a parallel implementation of the `AINV` algorithm combined with a graph partitioning precomputation, published in [5], indicate, that the `AINV` method can efficiently be implemented in parallel environments.

## 6.7 Approximate Inverses of Triangular Factors

In [46], Kolotilina and Yeremin propose methods for approximating the inverses of the triangular factors of Cholesky- and $LU$-factorizations on fixed matrix-patterns

for matrices in $\mathbb{R}^{n\times n}$. For a symmetric positive definite matrix $A$ with the Cholesky-factorization $A = L^T L$, the inverse $L^{-1}$ of the upper triangular matrix $L$ is approximated. For non-symmetric $A$, the inverses $L^{-1}$ and $U^{-1}$ of the $LU$-decomposition $A = LU$ are approximated. Importantly, no information on the original triangular factors $L$ and $U$ is necessary for determining these approximate inverses of the triangular factors. The resulting algorithms are referred to as "`FSAI`" in the relevant literature, in abbreviation of "Factorized Sparse Approximate Inverse". Some numerical results of `FSAI` implementations for fixed matrix-patterns of the approximate inverses are reported by Benzi and Tůma in [11] and by Benzi, Kouhia and Tůma in [4].
In this section, we briefly sketch the approaches as proposed by Kolotilina and Yeremin in [46]. Further, we fit these approaches into the framework of projection methods. Thereby, we not only obtain the approximation properties from chapter 4 for these methods, but too, by combining these methods with the adaptive pattern derivation strategies from chapter 5, we obtain two new preconditioning algorithms, namely the $\mathtt{L^TL}$`-projection` algorithm and the `LU-projection` algorithm. These new algorithms determine approximate inverses on adaptively generated matrix-patterns of the triangular factorization of the coefficient matrix. Importantly, the resulting algorithms, $\mathtt{L^TL}$`-projection` and `LU-projection` are highly parallel, and have potential for vectorization, and are thus well suited for the implementation on supercomputers. The numerical results of these methods (as reported in chapter 7) indicate that the adaptive pattern derivation substantially enhances both the robustness and the efficiency of these preconditioning methods, as compared with the results obtained by these methods for fixed a priori matrix-patterns.

### 6.7.1 The Hermitian Positive Definite Case

We begin this section with a brief summary of the results presented by Kolotilina and Yeremin in [46].

**The Basic Idea**
Let the matrix $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite and let $A = L^T L$ be the Cholesky-decomposition of $A$. Then an approximate inverse $P$ of the upper triangular matrix $L$ is sought. The basic idea for finding an approximate inverse $P$ of the Cholesky-factor $L$ rests on the equation

$$A = L^T L \Longleftrightarrow \quad A \underbrace{L^{-1}}_{\approx P} = L^T. \tag{6.39}$$

For the construction of an approximate inverse $P$ of the upper triangular factor $L$, an upper triangular matrix-pattern $S_P$ (according to definition 4.1 on page 43) for $P$ must be chosen in advance.

Let the columns of this matrix-pattern $S_P$ be denoted by $J_k$ for $k = 1, \ldots, n$. Then, the entries of the approximate inverse $P$ of the triangular factor $L$ are set by

$$(AP)_{ik} = L^T_{ik} \tag{6.40}$$

for $i \in J_k$ and $k = 1, \ldots, n$. Since the matrix $L^T$ is lower triangular and an upper triangular projective approximate inverse $P$ of $L$ is sought, equation (6.40) simplifies to

$$(AP)_{ik} = \begin{cases} l_{kk} & \text{for } i = k, \\ 0 & \text{else,} \end{cases} \tag{6.41}$$

for $i \in J_k$ and $k = 1, \ldots, n$, where $l_{kk}$ denotes the $k$-th diagonal entry of the Cholesky-factor $L$.

Thus, except of the scaling factors $l_{kk}$ for $k = 1, \ldots, n$, the inverse of the Cholesky-factor $L$ can be approximated on the pattern $S_P$ without using any information of the actual entries in $L$.

Since the diagonal elements $l_{kk}$, for $k = 1, \ldots, n$, of the Cholesky-factor $L$ are unknown in practice, the equation

$$\left(A\tilde{P}\right)_{ik} = \begin{cases} 1 & \text{for } i = k, \\ 0 & \text{else,} \end{cases} \tag{6.42}$$

where $i \in J_k$ and $k = 1, \ldots, n$, is utilized for calculating the approximate inverse $\tilde{P}$ of the Cholesky-factor $L$, and then diagonal scaling is applied to the obtained approximate inverse $\tilde{P}$ with the diagonal scaling matrix $D$ determined in such a way that

$$(D\tilde{P}^T A\tilde{P}D)_{ii} = 1,$$

for $i = 1, \ldots, n$.

Once such an approximate inverse $\tilde{P}D$ is computed, preconditioning can be performed by

$$A \longmapsto D\tilde{P}^T A\tilde{P}D.$$

Note that in this situation preconditioning from both sides should be applied, since then the coefficient matrix of the preconditioned linear system preserves the symmetry and the definiteness of the original coefficient matrix $A$.

Some statements on the approximation of sparse approximate inverses determined with the above approach to the true inverse are found in [46]. Further, we present a statement on this matter in corollary 6.8.

A severe drawback for the resulting FSAI algorithm proposed in [46] is the necessity of prescribing a matrix-pattern for the approximate inverse in advance, since in practice it is not known a priori what this matrix-pattern should be like.
In [24], the numerical properties of the FSAI preconditioner are surveyed. Further, in [24] the effect of reorderings of the coefficient matrix and some strategies for selecting appropriate sparsity patterns for the FSAI approach are discussed. In [25], results of a parallel implementation of the FSAI algorithm are presented.
In the following, we fit the FSAI preconditioning technique into the framework of projection methods. This provides both the theoretical approximation statements from chapter 4 and the adaptive pattern derivation techniques from chapter 5 for this preconditioning approach.

**The $\mathtt{L^TL}$-projection Method**
The formulation of the FSAI preconditioning technique proposed in [46] in terms of projection methods is straightforward:
Let the matrix $A \in I\!K^{n\times n}$ be Hermitian positive definite, and let $A = L^H L$ be the Cholesky-factorization of $A$.
We consider equation (6.40) with the notation $\tilde{P}$ instead of $P$, for both real and complex matrices:

$$\begin{aligned}
& \left(A\tilde{P}\right)_{ik} && = L^H_{ik} \\
\iff\ & u_i^H \left[A\tilde{P} - L^H\right] u_k && = 0 \\
\iff\ & u_i^H L^H \left[L\tilde{P} - I\right] u_k && = 0 \\
\iff\ & u_i^H L^H D \left[\hat{L}\tilde{P} - D^{-1}\right] u_k && = 0 \\
\iff\ & u_i^H L^H D \left[\hat{L}P - I\right] D^{-1} u_k && = 0
\end{aligned} \tag{6.43}$$

for $i \in J_k$, $k = 1, \ldots, n$, where the diagonal scaling matrix $D$ is chosen in such a way that the matrix $L^H D$ is unit upper triangular, and the matrix $P$ is defined by $P := \tilde{P}D$. Introducing the scaling matrix $D$ in the above equations corresponds to switching over from (6.41) to the simplified approach (6.42) in the method proposed by Kolotilina and Yeremin. With the scaling matrix $D$ in (6.43), the left-hand side projection matrix $Z_L = D^H L$ is unit upper triangular, and with that we obtain the explicit representation of the triangular projective approximate inverse $P$ in the form given in corollary 6.7.
By (6.43), the method for approximating the inverse of a triangular factor $L$ from [46] is a projection method for approximating the scaled upper triangular Cholesky-factor $\hat{L}$, where the projection matrices are $Z_L := D^H L$ and $Z_R := D^{-1}$ and the projection pattern $S_P = (J_1, \ldots, J_n)$ is upper triangular. In the following, we refer

to the projection method (6.43) by `LTL-projection` method.

By applying corollary 4.5 to the projection method (6.43), we obtain the explicit representation for the columns of the upper triangular projective approximate inverse $P$:

**Corollary 6.7** (Explicit Representation of the Triangular Projective Approximate Inverse)
*Let the matrix $A \in I\!\!K^{n\times n}$ be Hermitian positive definite and let $A = L^H L$ be the Cholesky-factorization of $A$. Let the projective approximate inverse $P$ of the upper triangular matrix $L$ be determined by the projection method (6.43) on the upper triangular projection pattern $(J_1, \dots, J_n)$. Then this projection method is explicit according to definition 4.3. If the relations $k \in J_k$ are satisfied for all $k = 1, \dots, n$, the projection method is practical according to definition 4.3. The columns $p_k$ of the projective approximate inverse $P$ have the unique explicit representation*

$$(p_k)_j = \begin{cases} \left([A(J_k, J_k)]^{-1} (u_k)(J_k)\right)_i & \text{for } j = (J_k)_i, \\ 0 & \text{else,} \end{cases} \tag{6.44}$$

*for $k = 1, \dots, n$.*

**Proof.**

The assertions follow directly from corollary 4.5 applied with $\hat{L}$ instead of $A$, with $Z_L := D^H$ and with $Z_R := D^{-1}$ by noting that

$$L^H D \hat{L} = L^H L = A$$

holds for the considered projection method and that the matrix $A$ is Hermitian positive definite.

$\diamondsuit$

From corollary 4.10, we obtain the following new statements on the quality of the approximation for the triangular projective approximate inverse $P$ given by the projection method (6.43) to the true inverse $L^{-1}$ of the Cholesky-factor $L$:

**Corollary 6.8** (Approximation Properties for the `LTL-projection` Method)
*Let the matrix $A \in I\!\!K^{n\times n}$ be Hermitian positive definite with the Cholesky-decomposition $A = L^H L$. We consider the `LTL-projection` method from (6.43), i.e. with $Z_L := D^H L$, $Z_R := D^{-1}$ and with some upper triangular projection pattern $(J_1, \dots, J_n)$. Let the columns of the upper triangular projective approximate inverse $P$ be denoted by $p_k$ for $k = 1, \dots, n$. Then, the identities*

$$\left\| \hat{L} p_k - u_k \right\|_{D^H D} = \min_{w \in I\!\!K^n_{J_k}} \left\| \hat{L} w - u_k \right\|_{D^H D} \tag{6.45}$$

*and*

$$\left\|p_k - \hat{L}^{-1}u_k\right\|_A = \min_{w \in I\!K^n_{J_k}} \left\|w - \hat{L}^{-1}u_k\right\|_A \tag{6.46}$$

*hold for* $k = 1, \ldots, n$.

**Proof.**

The assertions follow directly from corollary 4.10 applied to the `LTL-projection` method, i.e. with $D^H L$ instead of $Z_L$ and with $\hat{L}$ instead of $A$ by noting that for the matrices $D$, $L$ and $\hat{L}$ from (6.43) the equivalence

$$D\hat{L} = L \Longleftrightarrow \quad \hat{L}^{-1} = L^{-1}D,$$

and hence the equalities

$$D^H L\hat{L}^{-1} = D^H D$$

and

$$\hat{L}^H D^H L = A$$

hold.

◇

Equations (6.45) and (6.46) of the above corollary state that the upper triangular projective approximate inverse $P$ obtained with the `LTL-projection` method from corollary 6.43 columnwise minimizes the $D^H D$-norm of the quantities $\hat{L}w - u_k$ among all vectors $w$ with the pattern $J_k$, and the $A$-energy norm of the quantities $p_k - \hat{L}^{-1}u_k$, respectively.

As suggested in [46], the final preconditioning matrix for the Hermitian linear system $Ax = b$ is obtained by applying some column scaling to the projective approximate inverse $P$ determined with the `LTL-projection` method from (6.43). Numerous scaling variants are possible. We confine ourselves to present one scaling variant only:

**Lemma 6.9** (The `LTL-projection` Method Preconditioner), ([46], pp. 47-48)
*Let the matrix* $A \in I\!K^{n\times n}$ *be Hermitian positive definite and let* $L^H L = A$ *be the Cholesky-decomposition of* $A$*. Further, let the matrix* $P$ *be the triangular projective approximate inverse obtained with the* `LTL-projection` *method from* (6.43) *on some triangular projection pattern* $(J_1, \ldots, J_n)$*, let the relation* $k \in J_k$ *hold for*

*$k = 1, \ldots, n$, and let the diagonal of $P$ be zero-free. Then, with the diagonal scaling matrix $C$ be defined by*

$$C := \begin{pmatrix} \frac{1}{\left(u_1^H P u_1\right)^2} & & \\ & \ddots & \\ & & \frac{1}{\left(u_n^H P u_n\right)^2} \end{pmatrix},$$

*for the triangular approximate inverse defined by*

$$P_L := PC, \tag{6.47}$$

*the identities*

$$\left|\left(P_L^H A P_L\right)_{kk}\right| = 1$$

*hold for $k = 1, \ldots, n$, i.e. the diagonal elements of the preconditioned matrix $P_L^H A P_L$ have a absolute value of one.*

**Proof.**

The proof for the real case in given in [46]. The following proof for the case $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ proceeds analogously:
For $1 \leq k \leq n$ arbitrary fixed we have:

$$\begin{aligned} \left(P_L^H A P_L\right)_{kk} &= \left|u_k^H P u_k\right|^{-1} u_k^H P^H A P u_k \\ &= \left|u_k^H P u_k\right|^{-1} \left(p_k\left(J_k\right)\right)^H A\left(J_k, J_k\right) p_k\left(J_k\right) \\ &\quad \text{since the } k\text{-th column } p_k \text{ of } P \text{ has the pattern } J_k \\ &= \left|u_k^H P u_k\right|^{-1} u_k^H P u_k, \end{aligned}$$

by corollary 6.7.

$\diamondsuit$

In the forthcoming formulation of the `LTL-projection` algorithm for approximating the inverse $L^{-1}$ of an upper triangular Cholesky-factor $L$, we determine the approximate inverse $P_L$ as defined in (6.47). The preconditioned linear system has then the form

$$P_L^H A P_L y = P_L^H b, \; P_L y = x.$$

We carry over the general considerations regarding the adaptive pattern derivation from chapter 5 for the `LTL-projection` method stated in (6.43).

**Adaptive Pattern Derivation for the `LTL-projection` Algorithm**

For the `FSAI` preconditioner, as proposed by Kolotilina and Yeremin in [46], it is suggested to prescribe a fixed sparsity-pattern for the approximate inverse $P \approx L^{-1}$ in advance. For some problems, e.g. for problems with a known structure, guessing an appropriate pattern may be possible.
In [4] and [11], some numerical results, obtained by selecting the pattern for the approximate inverse equal to that of the original coefficient matrix $A$, are given. These results indicate that this choice for the pattern of the approximate inverse $P_L$, at last in general, is not advisable. The numerical results given in chapter 7 indicate that deriving the pattern of the approximate inverse $P_L$ adaptively produces substantially more robust and more efficient preconditioners.
Since the approach of Kolotilina and Yeremin given in [46] is, by (6.43), a projection method, the pattern of the triangular approximate inverse can be determined adaptively with the univariate and the multivariate pattern derivation algorithms introduced in chapter 5.

The $J_k$-residuals and the $J_k$-errors from definition 4.6 for the `LTL-projection` method from (6.43) are

$$r_{J_k} = \frac{1}{\delta_k}\left(\hat{L}p_k - u_k\right)$$

and

$$e_{J_k} = \frac{1}{\delta_k}\left(p_k - \hat{L}^{-1}u_k\right)$$

where $\delta_k$ denotes the $k$-th diagonal entry of the matrix $D$ from (6.43).
Since the scaled Cholesky-factor $\hat{L}$ as well as its inverse $\hat{L}^{-1}$ and the matrix $D$ are unknown in general applications, neither the $J_k$-residuals nor the $J_k$-errors are known explicitly. Thus, the $||.||_{D^H D}$-norm of the $J_k$-residuals cannot be used for the qualitative control of the pattern derivation. Nevertheless, the pattern derivation strategies from chapter 5 can be applied with the quantitative stopping criterion (see pages 67–68 for the stopping-criteria of the pattern derivation), since multiples of the decrease rates are known.

The following corollary gives the representation of the decrease rates $\lambda_j$ from definition (5.1) and of the univariate decrease rates $\theta_j$ from definition 5.7 for the `LTL-projection` method defined by (6.43).

**Corollary 6.10** (Decrease Rates for the `LTL-projection` Method)
*Let the matrix $A \in \mathbb{K}^{n\times n}$ be Hermitian positive definite, and let $A = L^H L$ be the Cholesky-factorization of $A$. Let the upper triangular projective approximate inverse $P$ of the scaled Cholseky factor $\hat{L}$ be determined with the* `LTL-projection` *method from* (6.43) *on some upper triangular projection pattern.*

*i) The univariate decrease rates from definition* 5.7 *have the form*

$$\theta_j = |\delta_k|^{-2} \frac{\left|u_j^H A p_k\right|^2}{||u_j||_A^2}, \tag{6.48}$$

*for* $1 \leq j < k$, $j \notin J_k$ *and* $k = 1, \ldots, n$, *where* $p_k$ *denotes the* $k$*-th column of* $P$.

*ii) The decrease rates from definition* 5.1 *are*

$$\lambda_j = |\delta_k|^{-2} \frac{\left|u_j^H A p_k\right|^2}{||u_j||_A^2 - g_j^H y_j}, \tag{6.49}$$

*for* $1 \leq j < k$, $j \notin J_k$ *and* $k = 1, \ldots, n$, *where* $p_k$ *denotes the* $k$*-th column of* $P$, *the vectors* $g_j \in I\!K^n_{J_k}$ *are defined by*

$$g_j := A\left(J_k, j\right),$$

*and the vectors* $y_j \in I\!K^n_{J_k}$ *are the solutions of the* $(\#J_k \times \#J_k)$ *linear systems*

$$A\left(J_k, J_k\right) y_j = g_j.$$

**Proof.**

We verify assertion $i$):
By applying lemma 5.8 with $\hat{L}$ instead of $A$, with $Z_L := D^H L$ and with $Z_R := D^{-1}$ we obtain

$$\theta_j = \frac{\left|u_j^H L^H D r_{J_k}\right|^2}{||u_j||^2_{L^H D \hat{L}}}$$

for $1 \leq j < k$, $j \notin J_k$ and $k = 1, \ldots, n$, where $p_k$ denotes the $k$-th column of $P$.
We consider the nominator:

$$\begin{aligned} u_j^H L^H D r_{J_k} =& \frac{1}{\delta_k} u_j^H L^H D \left(\hat{L} p_k - u_k\right) \\ =& \frac{1}{\delta_k} u_j^H A p_k - u_j^H L^H u_k, \end{aligned}$$

and since $u_j^H L^H u_k = 0$ for $1 \leq j < k$, we obtain the nominator in the form stated in (6.48). The denominator is obtained by noting that $L^H D \hat{L} = L^H L = A$.
Assertion $ii$) is obtained analogously by applying theorem 5.5 with $\hat{L}$ instead of $A$, $Z_L := L^H D$ and with $Z_R := D^{-1}$.

$\diamondsuit$

The decrease rates as shown in the above corollary are unknown, since the diagonal elements $\delta_k$ of the matrix $D$ from (6.43) are unknown in general applications. However, for the pattern derivation process it is sufficient to know the multiples $|\delta_k|^2\,\theta_j$ and $|\delta_k|^2\,\lambda_j$ for $k = 1, \ldots, n$ of the decrease rates. For notational convenience we define with $\theta_j$ from (6.48)

$$\tilde{\theta}_j := |\delta_k|^2\,\theta_j \tag{6.50}$$

and with $\lambda_j$ from (6.49)

$$\tilde{\lambda}_j := |\delta_k|^2\,\lambda_j. \tag{6.51}$$

**The `LᵀL-projection` Algorithm**

We summarize the above considerations by giving a pseudo-code formulation of the $\mathtt{L^TL\text{-}projection}$ algorithm (algorithm 15) for calculating the approximate inverse on an upper triangular Cholesky-factor.

The input data and parameters for the $\mathtt{L^TL\text{-}projection}$ algorithm (algorithm 15) are the same as for algorithm 8 (see pages 67–69), except of the parameter $\epsilon_k$, which is not needed since only the quantitative stopping criterion can be applied.

Note that the initial matrix-pattern $(J_1^0, \ldots, J_n^0)$ – apart from being upper triangular – must, as stated by corollary 6.7, satisfy $k \in J_k^0$ for $k = 1, \ldots, n$ such that the $\mathtt{L^TL\text{-}projection}$ method is practical according to definition 4.3.

On output the $\mathtt{L^TL\text{-}projection}$ algorithm (algorithm 15) furnishes the upper triangular approximate inverse $P_L$ of the Cholesky-factor $L$, where $A = L^H L$ is the Cholesky-factorization of the Hermitian positive definite matrix $A$.

In each step of the $k$-loop in step (1) of the $\mathtt{L^TL\text{-}projection}$ algorithm (algorithm 15), one column of the projective approximate inverse $P_L$ as defined in (6.47) on the adaptively derived pattern is determined. In step (1a) of the $\mathtt{L^TL\text{-}projection}$ algorithm, the $k$-th column of the projective approximate inverse $P$ given by the $\mathtt{L^TL\text{-}projection}$ method on the initial column-pattern is calculated. In the $l$-loop in step (1b) the pattern adaption is made. This begins with controlling the quantitative stopping criterion in steps (1(b)i) and (1(b)ii). If the stopping criterion is not satisfied, the set $L_{J_k}$, which contains the positive candidate indices, is determined in step (1(b)iii). In step (1(b)iv), the scaled decrease rates $\tilde{\theta}_j$ from (6.50) are calculated for all indices $j \in L_{J_k}$. The largest among these decrease rates are determined in step (1(b)v) and the corresponding indices are added to the current column-pattern in step (1(b)vi). In step (1(b)vii), the $k$-th column $p_k^l$ of the projective approximate inverse $P^l$ on the augmented column-pattern is calculated. When the pattern derivation for the current column is terminated, the $k$-th column $P_L u_k$ of the projective approximate inverse $P_L$ defined in (6.47) is obtained by scaling the corresponding final column $p_k^l$ of $P^L$.

1. For $k = 1, \ldots, n$

   (a) Compute the $k$-th column $p_k^0$ of the projective approximate inverse $P^0$ of the `LᵀL-projection` method (6.43) on the initial column-pattern $J_k^0$ by applying corollary 6.7

   (b) For $l = 1, \ldots, ms$

      i. $mt := \min\left(mfps, mf - \left(\# J_k^{l-1}\right)\right)$

      ii. If $(mt == 0)$ Exit $l$-loop

      iii. Determine the set $L_{J_k^{l-1}}$ defined by

      $$L_{J_k^{l-1}} := \left\{ j \notin J_k^{l-1} \mid 1 \leq j < k \;,\; \left(Ap_k^{l-1}\right)_j \neq 0 \right\}$$

      iv. Determine the decrease rates $\tilde{\theta}_j$ from (6.50) for all $j \in L_{J_k^{l-1}}$

      v. Determine the $mt$ indices $j_1, \ldots, j_{mt}$ according to the largest decrease rates $\tilde{\delta}_j$

      vi. $J_k^l := J_k^{l-1} \cup (j_1, \ldots, j_{mt})$

      vii. Compute the $k$-th column $p_k^l$ of the projective approximate inverse $P^l$ of the `LᵀL-projection` method (6.43) on the column-pattern $J_k^l$ by applying corollary 6.7

   (c) Obtain the $k$-th column $P_L u_k$ of the triangular approximate inverse $P_L$ by

   $$P_L u_k := \left(u_k^H p_k^l\right)^{-1/2} p_k^l$$

**Algorithm 15:** `LᵀL-projection`

Importantly, since the $k$-loop in step (1) of the `LᵀL-projection` algorithm can be done in parallel, the `LᵀL-projection` algorithm is well-suited for parallel computers.

**Computational Complexity of the `LᵀL-projection` Algorithm**
The steps that contribute to the computational complexity are:

**steps** (1a), (1(b)vii) Here, the columns $p_k^l$ of the projective approximate inverse $P$ are calculated on the corresponding column-patterns by invoking algorithm 7. By lemma 6.7, this involves the solution of linear systems with the coefficient matrix $A(J_k, J_k)$ for $1 \leq l \leq ms$ and for $k = 1, \ldots, n$. Step (1a) is done only once, whereas step (1(b)vii) is done at most $ms$ times.

**step** (1(b)iii) For determining the set $L_{J_k^{l-1}}$, the $\left(k - \#J_k^l - 1\right)$ sparse dot-products $u_i^H A\tilde{p}_k^{l-1}$ are computed.

**step** (1(b)iv) In this step, the scaled decrease rates $\tilde{\theta}_j$ are calculated according to (6.50). This step costs only one floating-point operation.
Alternatively, the scaled decrease rates $\tilde{\lambda}_j$ can be utilzed. In this case, for each index $j \in L_{J_k}$ a supplemental linear system with the coefficient matrix $A(J_k, J_k)$ must be solved and a sparse dot-product must be formed. Since this dot-product involves the vectors $A(J_k, j)$ for each $j \in L_{J_k}$, the resulting communication overhead on shared memory computers may be prohibitively high.

**step** (1c) For calculating the column $P_L u_k$ of the projective approximate inverse $P_L$, the finally obtained columns $p_k^l$ of $P$ are scaled.

Although not covered by the theoretical results deduced in this section, we apply the `LᵀL-projection` algorithm in our numerical tests (see chapter 7) to matrices which are symmetric, but not necessarily positive definite. The results of these tests, given in chapter 7, indicate that the `LᵀL-projection` algorithm is not only superior to the original `FSAI` approach from [46], but also competative to the state-of-the-art preconditioning methods for symmetric linear systems.

### 6.7.2 The Non-Hermitian Case

In this section, we consider the `FSAI` preconditioning technique proposed by Kolotilina and Yeremin in [46] for non-symmetric linear systems. This method is the generalization of the method discussed in section 6.7.1 to the non-symmetric case. For a non-symmetric matrix $A \in I\!R^{n\times n}$, which has the $LU$-decomposition $A = LU$, the inverses $L^{-1}$ and $U^{-1}$ of the triangular factors $L$ and $U$ are approximated on fixed matrix-patterns.
We classify the `FSAI` preconditioning technique for non-symmetric matrices in terms of projection methods, and thereby provide the approximation statements from section 4.2 and the pattern adaption strategies from chapter 5 for this approach. In particular, by combining the `FSAI` approach from [46] with the adaptive pattern derivation strategies developed in chapter 5, we obtain a new preconditioning algorithm – the `LU-projection` algorithm, which adaptively determines triangular approximate inverses, and their sparsity patterns, of the $LU$-decomposition $A = LU$ of the matrix $A$.

**The Basic Idea**
In analogy to (6.40), the inverses $P_L$ of $L$ and $P_U$ of $U$ are approximated by consid-

ering the following equations:

$$A = LU \iff A\underbrace{U^{-1}}_{\approx P_U} = L \iff \underbrace{L^{-1}}_{\approx P_L}A = U. \tag{6.52}$$

For the construction of $P_L$ and $P_U$, a lower triangular matrix-pattern $\left(\tilde{I}_1, \ldots, \tilde{I}_n\right)$ for $P_L$ and an upper triangular matrix-pattern $(J_1, \ldots, J_n)$ for $P_U$ must be prescribed in advance. The entries of the approximate inverses $P_L$ and $P_U$ are computed by

$$(P_L A)_{ik} = \begin{cases} u_{kk} & \text{i = k,} \\ 0 & \text{else,} \end{cases} \tag{6.53}$$

and by

$$(AP_U)_{lm} = \begin{cases} 1 & \text{l = m,} \\ 0 & \text{else,} \end{cases} \tag{6.54}$$

for $i \in \tilde{I}_k$, $l \in J_m$, and $k, m = 1, \ldots, n$, where $u_{kk}$ denotes the $k$-th diagonal entry of the matrix $U$. Note that – in analogy to the Hermitian positive definite case (see equation (6.41)) – in the above equations the triangularity of the matrices $L$ and $U$ is exploited.
The approximate inverse $P_U$ can directly be calculated by (6.54) on the prescribed matrix-pattern.
In general applications, the diagonal entries $u_{kk}$ of the upper triangular matrix $U$ are unknown. Thus, for computing the approximate inverse $P_L$ of the lower triangular matrix $L$ instead of equation (6.53), the simplified approach

$$\left(\tilde{P}_L A\right)_{ik} = \begin{cases} 1 & \text{for } i = k, \\ 0 & \text{else,} \end{cases} \tag{6.55}$$

for $i \in \tilde{I}_k$ and $k = 1, \ldots, n$, is applied, and some diagonal scaling is applied to the preconditioned linear system $P_L A P_U y = P_L b$, $PUy = x$, i.e. the preconditioned linear system

$$D_1 \tilde{P}_L A P_U D_2 y = D_1 \tilde{P}_L b, \; P_U D_2 y = x \tag{6.56}$$

is considered for the iterative solution process, where $D_1$, $D_2$ are some diagonal scaling matrices.

**The `LU-projection` Method**

We give the formulation of the above preconditioning technique in terms of projection methods for both real and complex linear systems:

Let the matrix-patterns $(I_1, \ldots, I_n)$ and $(J_1, \ldots, J_n)$ be upper triangular. We deduce the projection method formulation of the approach proposed by Kolotilina and Yeremin by considering equations (6.53) and (6.54).
Equation (6.54) is equivalent to

$$u_l^H \left[ AP_U - L \right] u_m = u_l^H L \left[ UP_U - I \right] u_m = 0, \tag{6.57}$$

for $l \in J_m$ and $m = 1, \ldots, n$, and equation (6.53) is equivalent to

$$\begin{aligned} & u_i^H \left[ A^H \tilde{P}_L^H - U^H \right] u_k && = 0 \\ \Longleftrightarrow\ & u_i^H U^H \left[ L^H \tilde{P}_L^H - I \right] u_k && = 0 \\ \Longleftrightarrow\ & u_i^H U^H D \left[ \hat{L}^H \tilde{P}_L^H - D^{-1} \right] u_k && = 0 \\ \Longleftrightarrow\ & u_i^H U^H D \left[ \hat{L}^H P_L^H - I \right] D^{-1} u_k && = 0, \end{aligned} \tag{6.58}$$

for $i \in I_k$ and $k = 1, \ldots, n$, where the diagonal scaling matrix $D$ is chosen in such a way that $U^H D$ is unit upper triangular. We define the matrix $\hat{L}^H$ by $\hat{L} := D^{-1} L^H$ and and $P_L^H$ by $P_L^H := \tilde{P}_L^H D$. The scaling matrix $D$ is used in (6.58), because for the explicit representation of the triangular projective approximate inverse $P_L^H$, as given by corollary 6.11, it is necessary that the left-hand side projection matrix is unit upper triangular. Introducing the scaling matrix $D$ in (6.58) corresponds to switching over from (6.53) to the simplified approach (6.55) in the approach of Kolotilina and Yeremin. Therefore, the projective approximate inverse $P_L$ determined by the projection method (6.58) approximates not the matrix $L$ itself, but the scaled matrix $\hat{L} = LD^{-H}$. Thus, as porposed by Kolotilina and Yeremin in (6.56), some scaling should by applied to the preconditioned linear system (this matter is considered on page 139).
Note that the projection method (6.58) actually is the transposed notation for a left-hand side projection method, i.e. the column-patterns $I_k$ actually are the patterns of the rows of the triangular projective approximate inverse $P_L$.

Altogether, by equations (6.57) and (6.58) the methods for approximating the inverses of the triangular factors $L$ and $U$ given in [46] are projection methods according to definition 4.2 with $Z_L := L^H$ and $Z_R := I$ on the projection pattern $(J_1, \ldots, J_n)$ for $P_U$, and with $Z_L := D^H U$ and $Z_R := D^{-1}$ on the projection pattern $(I_1, \ldots, I_n)$ for $P_L^H$.

We apply corollary 4.5 to the projection methods (6.57) and (6.58) in order to obtain the explicit representation of the upper triangular projective approximate inverses $P_U$ and $P_L^H$:

**Corollary 6.11** (Explicit Representation of the Triangular Projective Approximate Inverses $P_L^H$ and $P_U$)
*Let the matrix $A \in I\!\!K^{n\times n}$ be non-singular, and let $A = LU$ be the LU-decomposition of $A$.*

*i) Let the projective approximate inverse $P_U$ of $U$ be determined by the projection method (6.57), i.e. with the projection matrices $Z_L := L^H$ and $Z_R := I$ on the upper triangular projection pattern $(J_1, \ldots, J_n)$. If this projection method is $J_m$-explicit according to definition 4.3 for some $m \in I\!\!N$, then the m-th column $P_U u_m$ of $P_U$ has the form*

$$(P_U u_m)_j = \begin{cases} \left([A(J_m, J_m)]^{-1} u_m(J_m)\right)_i & \text{for } j = (J_m)_i, \\ 0 & \text{else.} \end{cases} \tag{6.59}$$

*Further, if this projection method is explicit, it is practical according to definition 4.3 if the relations $m \in J_m$ hold for $m = 1, \ldots, n$.*

*ii) Let the projective approximate inverse $P_L^H$ of $L^H$ be determined by the projection method (6.58), i.e. with the projection matrices $Z_L := D^H U$ and $Z_R := D^{-1}$ on the upper triangular projection pattern $(I_1, \ldots, I_n)$. If this projection method is $I_k$-explicit according to definition 4.3, then the k-column $P_L^H u_k$ of $P_L$ has the form*

$$\left(P_L^H u_k\right)_j = \begin{cases} \left([A(I_k, I_k)]^{-H} u_k(I_k)\right)_l & \text{for } j = (I_k)_l, \\ 0 & \text{else.} \end{cases} \tag{6.60}$$

*This projection method is practical according to definition 4.3, if it is explicit and if the relations $k \in I_k$ hold for $k = 1, \ldots, n$.*

**Proof.**

Assertion $i$) follows directly from corollary 4.5 applied to the projection method (6.57), i.e. with $U$ instead of $A$, $Z_L := L^H$ and with $Z_R := I$, by noting that $(Lu_m)(J_m) = u_m(J_m)$ since the matrix $L$ is unit lower triangular.
Assertion $ii$) follows directly from corollary 4.5 applied to the projection method (6.58), i.e. with $\hat{L}^H$ instead of $A$, $Z_L := D^H U$ and with $Z_R := D^{-1}$, by noting that $U^H D\hat{L} = U^H L^H = A^H$ and that $\left(U^H D u_k\right)(I_k) = u_k(I_k)$ since the matrix $U^H D$ is unit lower triangular.

$\diamondsuit$

By applying corollary 4.10 to the projection methods (6.57) and (6.58), we obtain new approximation properties for the triangular projective approximate inverses $P_L^T$ and $P_U$ to the exact inverses $\hat{L}^{-1}$ and $U^{-1}$. Note that in the following corollary only the positive real case from corollary 4.10 is stated. The case that $L^H U^{-1}$ or $D^H U \hat{L}^{-H}$ is Hermitian positive definite is irrelevant, since this would imply that the matrix $A$ is Hermitian positive definite as well. In this case, the considerations regarding the Cholesky-factorization of $A$ presented in section 6.7.1 apply. Thus, in the remainder of this section we consider the situation $\mathbb{K} = \mathbb{R}$ and $A \in \mathbb{R}^{n \times n}$ only. Further, we consider the univariate decrease rates $\theta_j$ from definition 5.7 for the adaptive pattern derivation only.

**Corollary 6.12** (Approximation Properties for $P_L$ and $P_U$)
*Let the matrix $A \in \mathbb{K}^{n \times n}$ be non-singular, and let $A = LU$ be the $LU$-decomposition of $A$.*

*i) Let the projective approximate inverse $P_U$ of $U$ be determined by the projection method (6.57), i.e. with the projection matrices $Z_L := L^T$ and $Z_R := I$ on the upper triangular projection pattern $(J_1, \ldots, J_n)$, and let this projection method be $J_m$-explicit according to definition 4.3 for some $1 \leq m \leq n$. Let the matrix $L^T U^{-1}$ be positive real, let $L^T U^{-1} = M + R$, where $M$ denotes the symmetric and $R$ denotes the skew-symmetric part of $L^T U^{-1}$. Let further $\rho(R)$ denote the spectral radius of $R$ and let $\mu_m(M)$ denote the minimal eigenvalue of $M$. Then the inequalities*

$$\left\| U P_U u_m - u_m \right\|_{L^T U^{-1}} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in \mathbb{R}^n_{J_m}} \left\| U w - u_m \right\|_{L^T U^{-1}} \tag{6.61}$$

*and*

$$\left\| P_U u_m - U^{-1} u_m \right\|_{A^T} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in \mathbb{R}^n_{J_m}} \left\| w - U^{-1} u_m \right\|_{A^T} \tag{6.62}$$

*hold.*

*ii) Let the projective approximate inverse $P_L^T$ of $\hat{L}^T$ be determined by the projection method (6.58), i.e. with the projection matrices $Z_L := D^T U$ and $Z_R := D^{-1}$ on the upper triangular projection pattern $(I_1, \ldots, I_n)$, and let this projection method be $I_k$-explicit according to definition 4.3 for some $1 \leq k \leq n$. Let $D^T U \hat{L}^{-T}$ be positive real, let $D^T U \hat{L}^{-T} = M + R$, where $M$ denotes the symmetric and $R$ denotes the skew-symmetric part of $D^T U \hat{L}^{-T}$. Let further*

*$\rho(R)$ denote the spectral radius of $R$ and let $\mu_m(M)$ denote the minimal eigenvalue of $M$. Then the inequalities*

$$\left\|\hat{L}^T P_l^T u_k - u_k\right\|_{D^T U \hat{L}^{-T}} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in I\!R^n_{I_k}} \left\|\hat{L}^T w - u_k\right\|_{D^T U \hat{L}^{-T}} \tag{6.63}$$

*and*

$$\left\|P_L^T u_k - \hat{L}^{-T} u_k\right\|_A \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2(M)}} \min_{w \in I\!R^n_{I_k}} \left\|w - \hat{L}^{-T} u_k\right\|_A \tag{6.64}$$

*hold.*

**Proof.**

The assertions follow directly from applying corollary 4.10 to the projection methods (6.57) and (6.58).

$\diamondsuit$

**Adaptive Pattern Derivation for the Approximate Inverses $P_L^T$ and $P_U$**
By the following considerations we carry over the pattern adaption strategies, deduced in chapter 5, to the projection methods (6.57) and (6.58).

The $J_m$-residuals $r_{J_m}^U$ and the $J_m$-errors $e_{J_m}^U$ from definition 4.6 for the projection method (6.57) have the form

$$r_{J_m}^U = U P_U u_m - u_m \tag{6.65}$$

and

$$e_{J_m}^U = P_U u_m - U^{-1} u_m$$

for $m = 1, \ldots, n$, and the $I_k$-residuals $r_{I_k}^L$ and the $I_k$-errors $e_{I_k}^L$ for the projection method (6.58) are

$$r_{I_k}^L = \frac{1}{\delta_k}\left(\hat{L}^T P_L^T u_k - u_k\right) \tag{6.66}$$

and

$$e_{I_k}^L = \frac{1}{\delta_k}\left(P_L^T u_k - \hat{L}^{-T} u_k\right)$$

for $k = 1, \ldots, n$, where $\delta_k$ denotes the $k$-th diagonal entry of the matrix $D$ from (6.58).
Since the matrices $\hat{L}$ and $U$ are unknown in practice, the $r^U_{J_k}$-residuals and the $r^L_{I_k}$-residuals as well as their $||.||_{L^T U^{-1}}$-norms and $||.||_{D^T U \hat{L}^{-T}}$-norms respectively, are unknown in general. Therefore, the adaptive pattern derivation process for the projection methods (6.57) and (6.58) cannot be controlled by the qualitative stopping criterion (see the discussion on pages 67–68 on that matter). Nevertheless, the pattern derivation strategies, as put forth in chapter 5, can be applied with the qualitative stopping criterion.

The following corollary gives explicit representations for the univariate decrease rates from definition 5.7 for the projection methods (6.57) and (6.58):

**Corollary 6.13** (Univariate Decrease Rates for $P_L^T$ and $P_U$)
*Let the non-singular matrix $A \in I\!R^{n\times n}$ have the LU-decomposition $A = LU$.*

*i) We consider the projection method* (6.57). *Let this projection method be $J_m$-explicit according to definition* 4.3 *for some* $1 \le m \le n$. *Further, let the matrix $L^T U^{-1}$ be positive real and let $M$ denote the symmetric part of $L^T U^{-1}$. Then the univariate decrease rates from definition* 5.7 *for the projection method* (6.57) *have the form*

$$\theta_j = \frac{\left(u_j^T U^T M r^U_{J_m}\right)^2}{\left\|u_j\right\|_A^2} \tag{6.67}$$

*for all elements $j$ of the candidate set $C_{J_m}$ from definition* 5.7, *where $r^U_{J_m}$ denotes the $J_m$-residuals from* (6.65).

*ii) We consider the projection method* (6.58). *Let this projection method be $I_k$-explicit according to definition* 4.3 *for some* $1 \le k \le n$. *Let the matrix $D^T U \hat{L}^{-T}$ be positive real and let $M$ denote the symmetric part of $D^T U \hat{L}^{-T}$. Then the univariate decrease rates from definition* 5.7 *for the projection method* (6.58) *have the form*

$$\theta_j = \frac{\left(u_j^T \hat{L} M r^L_{I_k}\right)^2}{\left\|u_j\right\|_A^2} \tag{6.68}$$

*for all elements $j$ of the candidate set $C_{I_k}$ from definition* 5.7, *where $r^L_{I_k}$ denotes the $I_k$-residuals from* (6.66).

**Proof.**

Assertion $i$) follows directly by applying corollary 5.8 to the projection method (6.57), i.e. with $U$ instead of $A$, with $Z_L := L^T$ and with $Z_R := I$.
Assertion $ii$) follows directly from applying corollary 5.8 to the projection method (6.58), i.e. with $\hat{L}^T$ instead of $A$, with $Z_L := D^T U$, with $Z_R := D^{-1}$ and by noting that

$$\left\|u_j\right\|_{A^T}^2 = \left\|u_j\right\|_A^2 .$$

$\diamondsuit$

**Approximate Univariate Decrease Rates**
The following considerations show that the nominators of the univariate decrease rates as shown in the above corollary are unknown in general applications:
First, we consider the nominator of equation (6.67), i.e. the representation of the univariate decrease rates for the projection method (6.57). For this projection method the matrix $M$ defined in assertion $i$) of corollary 6.13 is

$$M = \frac{1}{2}\left(L^T U^{-1} + U^{-T} L\right).$$

Hence, the quantity in the brackets of the nominator of the univariate decrease rates in (6.67) is

$$\begin{aligned} u_j^T U^T M r_{J_m}^U =& \frac{1}{2} u_j^T U^T \left(L^T U^{-1} + U^{-T} L\right)\left(U P_U u_m - u_m\right) \\ =& \frac{1}{2}\Big(\underbrace{u_j^T \left(A^T P_U + A P_U\right) u_m}_{\text{known}} - \underbrace{u_j^T A^T U^{-1} u_m}_{\text{unknown}} - \underbrace{u_j^T L u_m}_{=0 \text{ for } j<m}\Big). \end{aligned}$$

Since the exact univariate decrease rates are unknown, they must be estimated in practical applications. In analogy to the discussion on pages 92–93, we propose the following two estimates for the univariate decrease rates for the projection method (6.57):

$$\tilde{\theta}_j^1 := \frac{\left(u_j^T \left(A^T P_U + A P_U\right) u_m\right)^2}{\left\|u_j\right\|_A^2} \tag{6.69}$$

and

$$\tilde{\theta}_j^2 := \frac{\left(u_j^T A P_U u_m\right)^2}{\left\|u_j\right\|_A^2}, \tag{6.70}$$

for $1 \leq j < m$ with $j \in C_{J_m}$ and for $m$ such that the projection method (6.57) is $J_m$-explicit according to definition 4.3. The estimates $\tilde{\theta}_j^1$ are obtained by simply neglecting the unknown addend, the estimates $\tilde{\theta}_j^2$ are obtained by substituting the matrix $U^{-1}$ by $P_U$ in the unknown term.

We consider the nominator of equation (6.68), i.e. the representation of the univariate decrease rates for the projection method (6.58):
The matrix $M$ defined in point *ii*) of corollary 6.13 has the form

$$M = \frac{1}{2}\left(D^T U \hat{L}^{-T} + \hat{L}^{-1} U^T D\right).$$

Thus, for the quantity in the brackets of the nominator of the univariate decrease rates in (6.68) we have

$$\begin{aligned} u_j^T L M r_{I_k}^L =& \frac{1}{2\delta_k} u_j^T \hat{L}\left(D^T U \hat{L}^{-T} + \hat{L}^{-1} U^T D\right)\left(\hat{L}^T P_L^T u_k - u_k\right) \\ =& \frac{1}{2\delta_k}\left(u_j^T \left(A P_L^T + A^T P_L^T\right) u_k - u_j^T A \hat{L}^{-T} u_k - u_j^T U^T D u_k\right) \\ =& \frac{1}{2\delta_k}\Big(\underbrace{u_j^T \left(A P_L^T + A^T P_L^T\right) u_k}_{\text{known}} - \underbrace{u_j^T A \hat{L}^{-T} u_k}_{\text{unknown}} - \underbrace{u_j^T U^T D u_k}_{=0 \text{ for } j<k}\Big). \end{aligned}$$

Hence, the exact univariate decrease rates are unknown in general. In analogy to the discussion on pages 92–93, we propose the following two estimates for the univariate decrease rates for the projection method (6.58):

$$\tilde{\theta}_j^3 := \frac{\left(u_j^T \left(A P_L^T + A^T P_L^T\right) u_k\right)^2}{\left\|u_j\right\|_A^2} \tag{6.71}$$

and

$$\tilde{\theta}_j^4 := \frac{\left(u_j^T A^T P_L^T u_k\right)^2}{\left\|u_j\right\|_A^2}, \tag{6.72}$$

for $1 \leq j < k$ with $j \in C_{I_k}$ and for $k$ such that the projection method (6.58) is $I_k$-explicit. The estimates $\tilde{\theta}_j^3$ are obtained by simply neglecting the unknown addend, and the estimates $\tilde{\theta}_j^4$ are obtained by substituting the matrix $\hat{L}^{-T}$ by $P_L^T$ in the unknown term.

**Scaling the Preconditioned Linear System**
As suggested in [46], once the triangular approximate inverses $P_L$ of $\hat{L}$ and $P_U$ of $U$ are computed by the projection methods (6.57) and (6.58), some scaling should be

applied to the preconditioned linear system, i.e. instead of applying the iterative solver to the preconditioned system

$$P_L A P_U y = P_L b, \;\; P_U y = x,$$

the iterative solver should by applied to the linear system

$$D_1 P_L A P_U D_2 y = D_1 P_L b, \;\; P_U D_2 y = x,$$

where $D_1$ and $D_2$ denote some diagonal matrices. Theoretically, any scaling variant is possible, e.g. by defining $D_2 := I$ and

$$D_1 := \begin{pmatrix} \left(u_1^T P_U u_1\right)^{-1} & & \\ & \ddots & \\ & & \left(u_n^T P_U u_n\right)^{-1} \end{pmatrix}, \tag{6.73}$$

or by choosing $D_1$ and $D_2$ in such a way that

$$\left(P_L A P_U\right)_{kk} = 1 \tag{6.74}$$

for $k = 1, \ldots, n$. We consider the latter scaling variant in greater detail: Theoretically, $2n$ sparse dot-products are necessary to calculate the scaling factors $\left(P_L A P_U\right)_{kk}^{-1}$ directly. The computational complexity of this strategy is prohibitively high in practical applications. The following lemma given in [46] considers a remedy for this problem for certain cases:

**Lemma 6.14** ([46], p. 49)
*Let the non-singular matrix $A$ have the LU-decomposition $A = LU$. Let the projective approximate inverses $P_L^T$ and $P_U$ be determined by the projection methods* (6.57) *and* (6.58) *respectively, on the upper triangular projection patterns $(J_1, \ldots, J_n)$ and $(I_1, \ldots, I_n)$. Further, let these projection methods be explicit according to definition* 4.3. *Then the identities*

$$\left(P_L A P_U\right)_{ii} = \begin{cases} \left(P_U\right)_{ii} & \text{if } I_i \subset J_i \\ \left(P_L\right)_{ii} & \text{if } J_i \subset I_i \end{cases} \tag{6.75}$$

*hold for $i = 1, \ldots, n$.*
*Thus, if $I_i = J_i$ for $i = 1, \ldots, n$, then*

$$\left(P_L A P_U\right)_{ii} = \left(P_L\right)_{ii} = \left(P_U\right)_{ii} \tag{6.76}$$

*holds for $i = 1, \ldots, n$.*

If the matrix-patterns of both $P_L$ and $P_U$ are determined adaptively in general practical applications, the above lemma possibly is not applicable, because none of the relations between the projection patterns of $P_L$ and $P_U$ from (6.75) is satisfied. As a variant of determining the matrix-patterns of both triangular projective approximate inverses $P_L^T$ and $P_U$ adaptively, only for one of these matrices the pattern is determined adaptively, while the other matrix is calculated without adaptive pattern derivation on the projection pattern of the first matrix. This strategy reduces the overall CPU-time, but the quality of the approximation of the latter matrix – in the sense of corollary 6.12 – possibly is diminished. Hence, for this strategy the convergence of the preconditioned iteration might be insufficiently accelerated. An advantage of this approach is, that lemma 6.14 can be applied for scaling the preconditioned linear system.

**The `LU-projection` Algorithm**

In algorithm 16, we give the pseudo-code of the `LU-projection` algorithm for computing the approximate inverses $P_L$ of $\hat{L}$ and $P_U$ of $U$ – where $A = LU$ is the $LU$-decomposition of $A$ – according to the projection methods (6.57) and (6.58) on adaptively derived projection patterns. Further, a scaling matrix $D$ is determined. The input data and the parameters for the `LU-projection` algorithm (algorithm 16) are the matrix $A$ and two sets – one for the projection method (6.57) and one for the projection method (6.58) – of the parameters that control the pattern derivation process as explained for algorithm 8 (see pages 67–69). Since the $J_m$-residuals $r^U_{J_m}$ and the $I_k$-residuals $r^L_{I_k}$ respectively, as stated in (6.65) and (6.66), are unknown in practical applications, only the quantitative stopping criterion for the pattern derivation process can be applied. For the parameters $ms$, $mt$, and $mfps$ we use the subscripts $L$ and $U$. The parameters for the projection method (6.57) are those with the $U$ subscripts, and the parameters with the $L$ subscripts are for the projection method (6.58).
Further, two initial patterns $(J_1^0, \ldots, J_n^0)$ (for the projection method (6.57)) and $(I_1^0, \ldots, I_n^0)$ (for the projection method (6.58)) must be supplied. Note that for practical applications both initial patterns must contain the main diagonal, i.e. $m \in J_m^0$ and $k \in I_k^0$ must be satisfied for $m, k = 1, \ldots, n$, since then by corollary 6.11 both projection methods, if the are explicit, are already practical according to definition 4.3.
On output, the two triangular projective approximate inverses $P_U$ and $P_L$ determined with the projection methods (6.57) and (6.58) on adaptively derived projection patterns and the scaling matrix $D$ are obtained.

After calculating the projective approximate inverses $P_L$, $P_U$ and the scaling matrix $D_1$ by the `LU-projection` algorithm (algorithm 16), the preconditioned linear system

$$D_1 P_L A P_U y = D_1 P_L b, \;\; P_U y = x$$

1. For $m = 1, \ldots, n$
   (a) Compute the column $p_m^0$ of the triangular projective approximate inverse $P_U^0$ on the initial column-pattern $J_m^0$ by lemma 6.11
   (b) For $l = 1, \ldots, ms_U$
      i. $mt_U := \min\left(mfps_U, mf_U - \left(\#J_m^{l-1}\right)\right)$
      ii. If $(mt_U == 0)$ Exit $l$-loop
      iii. Determine the set $L^U_{J_m^{l-1}}$ defined by
      $$L^U_{J_m^{l-1}} := \left\{ j \notin J_m^{l-1} \mid 1 \le j < m,\ u_j^T A P_U u_m \neq 0 \right\}$$
      iv. Determine the decrease rates $\tilde{\theta}_j^2$ from (6.70) for all $j \in L^U_{J_m^{l-1}}$
      v. Determine the $mt_U$ indices $j_1, \ldots, j_{mt_U}$ according to the largest decrease rates $\tilde{\theta}_j^2$
      vi. $J_m^l := J_m^{l-1} \cup (j_1, \ldots, j_{mt_U})$
      vii. Compute the $m$-th column of the triangular projective approximate inverse $P_U^l$ on the column-pattern $J_m^l$ by lemma 6.11
2. For $k = 1, \ldots, n$
   (a) Compute the $k$-th column $(p_k^0)^T$ of the triangular projective approximate inverse $(P_L^0)^T$ on the initial column-pattern $I_k^0$ by lemma 6.11
   (b) For $l = 1, \ldots, ms_L$
      i. $mt_L := \min\left(mfps_L, mf_L - \left(\#I_k^{l-1}\right)\right)$
      ii. If $(mt_L == 0)$ Exit $l$-loop
      iii. Determine the set $L^L_{I_k^{l-1}}$ defined by
      $$L^L_{I_k^{l-1}} := \left\{ j \notin I_k^{l-1} \mid 1 \le j < k,\ u_j^T A^T P_L^T u_k \neq 0 \right\}$$
      iv. Determine the decrease rates $\tilde{\theta}_j^4$ from (6.72) for all $j \in L^L_{I_k^{l-1}}$
      v. Determine the $mt_L$ indices $j_1, \ldots, j_{mt_U}$ according to the largest decrease rates $\tilde{\theta}_j^4$
      vi. $I_k^l := I_k^{l-1} \cup (j_1, \ldots, j_{mt_U})$
      vii. Compute the $k$-th column $(p_k^0)^T$ of the triangular projective approximate inverse $(P_L^0)^T$ on the column-pattern $I_k^l$ by lemma 6.11
3. Determine the scaling matrix $D_1$ according to (6.73)

**Algorithm 16:** `LU-projection`

is handed over to the iterative solver. Alternatively, the one-sided preconditionings

$$AP_U D_1 P_L y = b,\ P_U D_1 P_L y = x$$

or

$$P_U D_1 P_L A x = P_U D_1 P_L b,$$

can be considered.

Alternatively to the estimates $\tilde{\theta}_j^2$ in step (1(b)iv) of the `LU-projection` algorithm (algorithm 16), the estimates $\tilde{\theta}_j^1$ from (6.69) may be applied. In this case the sets $L^U_{J_m^{l-1}}$ from step (1(b)iii) are defined by

$$L^U_{J_m^{l-1}} := \left\{ j \notin J_m^{l-1} \mid 1 \le j < m,\ u_j^T \left( A^T P_U + A P_U \right) u_m \neq 0 \right\}.$$

Analogously, instead of the estimates $\tilde{\theta}_j^4$ in step (2(b)iv) of the `LU-projection` algorithm (algorithm 16), the estimates $\tilde{\theta}_j^3$ can be utilized. The corresponding sets $L^L_{I_k^{l-1}}$ is step (2(b)iii) have then the form

$$L^L_{I_k^{l-1}} := \left\{ \notin I_k^{l-1} \mid 1 \le j < k,\ u_j^T \left( A P_L^T + A^T P_L^T \right) u_k \neq 0 \right\}.$$

Instead of the scaling in (3), any other scaling can be applied.

The `LU-projection` algorithm consists of three major sections:
In the $m$-loop in step (1) the projective approximate inverse $P_U$ from projection method (6.57) is determined.
The projective approximate inverse $P_L^T$ from projection method (6.58) is determined in the $k$-loop in step (2).
In step (3) the scaling matrix $D$ is determined.

Because steps (1) and (2) of the `LU-projection` algorithm correspond to algorithm 9 with the quantitative stopping criterion for the projection methods (6.57) and (6.58), the `LU-projection` algorithm is inherently parallel, and thus well-suited for today's supercomputers.

**Computational Complexity of the `LU-projection` Algorithm**
For an explanation of the algorithmic flow and of the general computational complexity of the `LU-projection` algorithm (algorithm 16) the remarks given to algorithm 9 on pages 73–75 apply seperately to steps (1) and (2).
We consider the computational cost of the estimates for the decrease rates $\tilde{\theta}_j^1$ from (6.69) and $\tilde{\theta}_j^2$ from (6.70) in greater detail. For the estimates $\tilde{\theta}_j^3$ from (6.71) and $\tilde{\theta}_j^4$ from (6.72) analogous statements hold.

**step** (1(b)iii) If the estimates $\tilde{\theta}_j^2$ are utilized for the pattern derivation, for determining the sets $L^U_{J_m^{l-1}}$ the sparse dot-product $u_j^T A P_U u_m$ needs to be formed for each $j \notin J_m^{l-1}$ and for $l = 0, \ldots, ms_U$.
If the estimates $\tilde{\theta}_j^1$ are considered for the pattern derivation, additionally the sparse dot-product $u_j^T A^T P_U u_m$ must be calculated for each $j \notin J_m^{l-1}$ and for $l = 1, \ldots, ms_U$. The sparsity of these dot-products depends essentially on the number of non-zeros in the corresponding columns $P_U u_m$ of $P_U$.
This step of the `LU-projection` algorithm may be a problem in a distributed computing environment, since forming these dot-products requires arbitrary access to the columns of the matrix $A$, and, if the $\tilde{\theta}_j^1$ are used, access to both arbitrary columns and rows of the matrix $A$. The resulting amount of interprocessor communication may diminish the scalability of the `LU-projection` algorithm.

**step** (1(b)iv) Since the nominators of the estimates $\tilde{\theta}_j^1$ or $\tilde{\theta}_j^2$ are already known from step (1(b)iii), the steps (1(b)iv) are almost cost-free. Only two floating-point operations are necessary for each $j \notin J_k$ and for $l = 0, \ldots, ms_U$.

Altogether, in this section we have fit the two `FSAI` preconditioning approaches given in [46] into the theoretical framework of projection methods. Importantly, by combining these approaches with the adaptive pattern derivation strategies developed in chapter 5, we obtain two new inherently parallel preconditioning algorithms, the `L`$^{\mathtt{T}}$`L-projection` algorithm and the `LU-projection` algorithm. These new algorithms are superior to the original `FSAI` approaches given in [46], and, as the results of the numerical tests presented in chapter 7 indicate, are competative to the state-of-the-art preconditioning techniques.

## 6.8 Summary and further Methods

In this section, for providing an overall view of the considered methods, we give a brief summary of the preconditioning techniques considered in this chapter. Further, we briefly sketch further preconditioning techniques to complete the picture.

In table 1, we draw up the preconditioning techniques considered in sections 6.1–6.7. In the column designated by "approximates" in table 1, we register which matrix is approximated by the corresponding preconditioning technique. The column designated by "principle" gives brief information on the construction principle of the considered preconditioning techiques. For the projection methods listed in table 1, the projection matrices $Z_L$ and $Z_R$ according to definition 4.2 and the possible decrease rates for the pattern derivation are given. Ample considerations on the methods contained in table 1 are found in the corresponding subsections. Except of the incomplete decomposition techniques (`ILU` and `IC`), all methods stated in table 1 are efficiently parallelizable.

| method | approximates | principle | section |
|---|---|---|---|
| Normalization | $A^{-1}$ | Projection method with $Z_L := I$ and $Z_R := I$, or $Z_R := A^H$ on a fixed diagonal projection pattern | 6.1 |
| `SPAI` | $A^{-1}$ | Projection method with $Z_L := A$ and $Z_R := I$ on a adaptively determined projection pattern, decrease rates: $\theta_j$, $\lambda_j$ | 6.2 |
| `Plain projection` | $A^{-1}$ | Projection method with $Z_L := Z_R := I$ on an adaptively determined projection pattern, decrease rates: $\theta_j$, $\lambda_j$ and estimates | 6.3 |
| `ILU` | $L$, $U$ | Incomplete decomposition $A = LU + E$, various strategies for fixed or adaptive patterns | 6.4 |
| `IC` | $L$ | Incomplete decomposition $A = L^T L + E$, various strategies for fixed or adaptive patterns | 6.5 |
| `AINV` | $L^{-1}$, $U^{-1}$, $D$ | Incomplete approximation of $A = LDU$, based on calculating two sets of $A$-biconjugate vectors, adaptive pattern | 6.6 |
| $\mathtt{L^TL}$`-projection` | $L^{-1}$ | Projection method with $Z_L := D^H L$ and $Z_R := D^{-1}$ on adaptive projection pattern, decr. rates: estimates of $\theta_j$, $\lambda_j$ | 6.7.1 |
| `FSAI` (symmetric) | $L^{-1}$ | Projection method with $Z_L := D^H L$ and $Z_R := D^{-1}$ on a fixed a priori projection pattern | 6.7.1 |
| `LU-projection` | $L^{-1}, U^{-1}$ | Projection methods with $Z_L := L^H$ and $Z_R := I$ for $P_U$ and with $Z_L := D^H U$ and $Z_R := D^{-1}$ on adaptive projection patterns, decr. rates: estimates of $\theta_j$ | 6.7.2 |
| `FSAI` (non-symmetric) | $L^{-1}, U^{-1}$ | Projection methods with $Z_L := L^H$ and $Z_R := I$ for $P_U$ and with $Z_L := D^H U$ and $Z_R := D^{-1}$ on fixed a priori projection patterns | 6.7.2 |

Table 1: The preconditioning techniques considered in sections 6.1–6.7

In the following, for the sake of completeness, we briefly sketch further preconditioning techniques.

**Incomplete $QR$-decomposition**
Analogously to the incomplete $LU$-decomposition, an incomplete $QR$-decomposition can be considered:

$$A = QR + E, \tag{6.77}$$

where $Q$ is orthonormal and $R$ is upper triangular. The entries $e_{ij}$ of the error-matrix $E$ are given by

$$e_{ij} = \begin{cases} 0 & \text{for } (i,j) \in S, \\ a_{ij} - \sum_{l=1}^{n} q_{il} r_{lj} & \text{for } (i,j) \notin S, \end{cases}$$

where $a_{ij}$, $q_{ij}$ and $r_{ij}$ denote the entries of the matrices $A$, $Q$ and $R$, and $S$ denotes an either prescribed or adaptively determined pattern. By neglecting the matrix $E$ and by viewing the matrix $QR$ as an approximation to $A$, preconditioning can by performed with

$$P_L = R^{-1} R^{-T} A^T, \; P_R = I$$

for left side preconditioning, with

$$P_L = I, \; P_R = R^{-1} R^{-T} A^T$$

for right-hand side preconditioning, or with

$$P_L = R^{-T} A^T, \; P_R = R^{-1}$$

for two-sided preconditioning. In each iteration step with the preconditioned system, two triangular systems have to be solved: one with $R$ and one with $R^T$. Thus, this preconditioning suffers from similar sequential bottlenecks as the `ILU` approach.
As a modification of the above `IQR`-approach, a decomposition of the form

$$AR = Q + E \tag{6.78}$$

can be computed. This decomposition leads to the following preconditioning matrices:

$$P_L = RR^T A^T, \; P_R = I$$

for right-hand side,

$$P_L = I,\ P_R = RR^T A^T$$

for left-hand side, and

$$P_L = R^T A^T,\ P_R = R$$

for two-sided preconditioning.
The advantage of this approach is that during the iterative solution no triangular systems need to be solved. Instead of this, three matrix-vector products are performed in each iteration step. Thus, once the matrix $R$ is computed, this approach is efficiently vectorizable and parallelizable. For investigations on the properties of `IQR` methods, we refer e.g. to [8], [36], [56] and [59].

**Polynomial Preconditioners**
Another family of preconditioning techniques, the "Polynomial Preconditioners" rests on the approach

$$P := \sum_{i=1}^{j} \mu_i A^i + \mu_0 I, \tag{6.79}$$

where $\mu_i \in I\!R$ for $i = 1, \ldots, n$. Various strategies for choosing $j$ and $\mu_i$ for $i = 1, \ldots, j$ exist, see e.g. [26] and [41]. For instance, $j$ and $\mu_i$ can be chosen in order to approximate the Neumann series: If $\rho\,(I - A) < 1$, then

$$A^{-1} = \sum_{i=1}^{\infty} (I - A)^i + I.$$

Another possibility is to choose $j$ and $\mu_i$ in such a way that

$$AP - I = \sum_{i=0}^{j} \mu_i A^{i+1} - I$$

is minimized. Polynomial preconditioners can be vectorized and parallelized, since they basically involve matrix-vector multiplications. However, at least in tendency, the efficiency and the robustness of polynomial preconditioners is inferior to the results obtained by other preconditioning techniques.

**Incomplete Gauss-Jordan**
With the Gauss-Jordan algorithm, the inverse $A^{-1}$ of a matrix $A$ is calculated via

a sequence of intermediate matrices:

$$\begin{aligned} A^{(0)} &:= A, \\ P^{(0)} &:= I, \\ A^{(i)} &:= C^{(i)} A^{(i-1)}, \\ P^{(i)} &:= C^{(i)} P^{(i-1)}, \end{aligned}$$

for $i = 1, \ldots, n$, where the matrices $C^{(i)}$ are chosen in such a way that the $i$-th column of the matrix $A^{(i)}$ is a multiple of the $i$-th unit-vector. The resulting matrix $A^{(n)}$ is a diagonal matrix, and the relations

$$\begin{aligned} A^{(n)} &:= C^{(n)} \cdot \cdots \cdot C^{(1)} A, \\ P^{(n)} &:= C^{(n)} \cdot \cdots \cdot C^{(1)}, \\ A^{-1} &:= A^{(n)-1} P^{(n)}, \end{aligned}$$

hold. With the incomplete Gauss-Jordan algorithm, a sequence of matrices $P^{(i)}$ and $A^{(i)}$ with a certain sparsity pattern is determined:

$$\begin{aligned} A^{(0)} &:= A, \\ P^{(0)} &:= I, \\ C^{(i)} &:= I + \left( I - \frac{1}{a_{ii}^{(i-1)}} A^{(i-1)} \right) u_i u_i^T, \\ a_{kl}^{(i)} &:= \begin{cases} \sum_{j=1}^n c_{kj}^{(i)} a_{jl}^{(i-1)} & \text{if } (k,j) \in S \\ 0 & \text{if } (k,j) \notin S \end{cases} \\ p_{kl}^{(i)} &:= \begin{cases} \sum_{j=1}^n c_{kj}^{(i)} p_{jl}^{(i-1)} & \text{if } (k,j) \in S \\ 0 & \text{if } (k,j) \notin S \end{cases}, \end{aligned}$$

for $i = 1, \ldots, n$, where $u_i$ denotes the $i$-th unit-vector, $a_{kl}^{(i)}$, $c_{kl}^{(i)}$ and $p_{kl}^{(i)}$ denote the $(k,l)$-th entries of the matrices $A^{(i)}$, $C^{(i)}$ and $P^{(i)}$ and $S$ denotes some matrix-pattern. Note that the above procedure guarantees that the matrix $A^{(n)}$ is a diagonal matrix. The preconditioning matrix $P$ is then defined by

$$P := A^{(n)-1} P^{(n)}. \tag{6.80}$$

The incomplete Gauss-Jordan algorithm is inherently recursive, and hence not efficiently parallelizable.

**Iterative Methods**
In principle, any iterative method can be applied to determine the columns $p_i$, for $i = 1, \ldots, n$, of an approximate inverse $P$ by carrying out a few iterations on the linear systems

$$Ap_i = u_i,$$

where $u_i$ denotes the $i$-th unit vector. Such approaches are considered e.g. in [11], [16] and [61]. The numerical results given in [11] indicate that such methods are not robust.

**Other Methods**
The use of wavelets for solving integral and differential equations is considered e.g. in [30]. In [15], the effect of transforming the inverse $A^{-1}$ into a wavelet basis prior to determining an approximate inverse is considered. The results given in [15] indicate, that such a wavelet transformation can enhance the efficiency and the robustness of methods for calculating approximate inverses.

The idea of multi-level methods for solving partial differential equations by the finite difference or by the finite element method is to consider a sequence of different discretizations. Various strategies constructing such sequences of grids are known (see e.g. [35] for an ample survey and [13], [63] for recent results). The idea behind these methods is, that obtaining a solution on a coarse grid is computationally cheap and may transfer much information onto finer grids.
Algebraic multigrid methods rest on considering the graph of the considered matrix. Instead of considering grids coarser and finer grids, the reduction process of algebraic multigrid considers relations between the elements contained in the matrix (see e.g. [31], [55]).

In [50], a family of Eirola-Nevanlinna methods (EN) is introduced. These methods combine the solver iteration with the calculation of an approximate inverse by rank-one updates in each iteration step. This approach is extended for linear systems with multiple right-hand sides in [51].

A preconditioning technique for symmetric positive definite matrices which is based on a new decomposition of the matrix is introduced in [43].

In [47], a preconditioning technique based on determining an approximate inverse of the skew-symmetric part of a matrix is surveyed.

In general, changing the physical model that leads to linear systems may be useful form of preconditioning, in the sense that the resulting linear systems can be solved more efficiently. For instance, in the context of partial differential equations, the iterative solution of the linear systems arising from discretization may be enhanced by changing the boundary conditions or by using FOSLS (First Order Systems Least Squares, see [14]).

# 7 Numerical Tests

In this chapter, we consider the numerical properties of the projection methods proposed in chapter 6, namely the `L`$^T$`L-projection` (see section 6.7.1), the `LU-projection` (see section 6.7.2) and the `Plain projection` (see section 6.3). In particular, we discuss the numerical properties of the different pattern derivation strategies derived for these projection methods in chapter 5.
Further, we compare the new projection methods to some state-of-the-art preconditioning methods described in chapter 6.

The iterative solvers considered are `CG` for symmetric problems, and `PRES20`, `BiCGstab` and `ATPRES` for unsymmetric problems (these iterative solvers are described in sections 3.5.1 – 3.5.4).
A description of the matrices included in our test set is given in section 7.2. The problems considered in our numerical tests are taken from the Harwell–Boeing collection [22], from the Cylshell collection, from the Hamm collection, from the Sparskit collection [2] and from the Tim Davis collection [3] (see [19] for details about these collections). Additionally, we consider some problems arising from the discretization of simplified models of the 3–dimensional Navier–Stokes equations and of the 2–dimensional Laplace equations.
Although many of the preconditioning techniques described in chapter 6 are suitable for both real and complex linear systems, for reasons of simplicity we confine ourselves to consider linear systems with real coefficient matrices only.
Because of the multitude of known preconditioning techniques, it is impossible to compare all of them in this work. Thus, we confine ourselves to consider only a few well known efficient methods. In particular, for the symmetric coefficient matrices in our test set, we compare the following preconditioning techniques:

- `IC(0)`: This method is the incomplete Choleskian decomposition as described in section 6.5, with the pattern $S$ being equal to the pattern of the upper triangular part of the original matrix. The resulting preconditioner is an approximation to the upper triangular factor of Choleskian decomposition of the original matrix. For this method, both the set-up time for the preconditioner and the preconditioned iterative solve are strongly sequential. Since the amount of non-zeros in the preconditioner is prescribed in advance, the required computer memory can be estimated in advance. No parameters are required for this method, which makes it easy to use, but on the other hand, it cannot be tuned, if the convergence of the iterative solve is insufficient.

[2] These matrices can be downloaded from http://math.nist.gov/MatrixMarket from the particular collection subdirectory.

[3] This collection can be accessed at http://www.cise.ufl.edu/~davis/sparse/.

- `AINV`: The preconditioner calculated by this method (this is algorithm 14, see section 6.6) for symmetric matrices is an approximation to the upper triangular factor of the inverse of the upper triangular Cholesky-factor of the original matrix. Thus, the preconditioned iteration can be efficiently implemented on parallel computers. For this method, one parameter, the dropping-threshold for the biconjugate vectors, must be supplied. The amount of computer memory required is not known in advance. Although the set-up time for the preconditioner is sequential in principle, an efficient parallel implementation of this method is described in [5].

- $\mathtt{L}^\mathtt{T}\mathtt{L}$`-projection`: This method (described in section 6.7.1, algorithm 15) adaptively approximates the inverse of the upper triangular Choleskian factor of the original matrix. Several parameters for controlling this algorithm must be supplied. The amount of computer memory required by this algorithm is known in advance. For this method, both the set-up time for the preconditioner and the preconditioned iteration can efficiently be implemented on parallel computers.

  As a variant, the $\mathtt{L}^\mathtt{T}\mathtt{L}$`-projection` can be applied with a fixed prescribed projection pattern. We consider for this variant projection patterns, which equal the upper triangle of the pattern of the original coefficient matrix. The resulting algorithm, in the relevant literature denoted by `FSAI`, coincides with the method proposed in [46]. The `FSAI` algorithm requires no parameters, and is thus easy to use, but, on the other hand, it cannot be tuned in case of unsatisfactory convergence of the preconditioned iteration. Both the set-up time and the iterative solve can efficiently be done in parallel.

For the non-symmetric matrices in our test set, we compare the following preconditioning techniques:

- `ILU(0)`: For this particular variant of the family of `ILU` methods (see section 6.4), fill-in in the triangular factors $L$ and $U$ is only allowed in locations corresponding to non-zeros in the original matrix. This method is parameter-free, hence easy to use, but not tunable. The required amount of computer memory for this method can be estimated in advance. Further, this approach is strongly sequential, both in the set-up phase and during the iterative solve, and thus not well-suited for the implementation on today's parallel computers.

- `SPAI`: As described in section 6.2, this method is a projection method. Hence, this method is inherently parallel, although a parallel implementation of this method is highly non-trivial due to a large amount of interprocessor communication when used with adaptive pattern derivation (see the discussion in section 6.2 for details on this matter). Depending on several parameters that have to be prescribed, the amount of computer memory required for this algorithm can be estimated in advance.

- `Plain projection`: Since this method is a projection method (described in section 6.3), it is inherently parallel (detailed comments on that matter are given after algorithm 13), and it is thus well suited for today's supercomputers. Several parameters must be supplied for this algorithm. The amount of computer memory for calculating the projective approximate inverse can be estimated in advance.

- `AINV`: This is algorithm 14 (see section 6.6). The preconditioner calculated by this method for unsymmetric matrices is an approximation to the inverses of the triangular factors $L$ and $U$, where $A = LU$ is the Gaussian decomposition of the matrix $A$. For this method, the remarks given to the `AINV` method for symmetric linear systems apply.

- `LU-projection`: This algorithm (algorithm 16, see section 6.7.2) consists of two projection methods; one for approximating the inverse $L^{-1}$ of the lower triangular matrix $L$, and one for approximating the inverse $U^{-1}$ of the upper triangular matrix $U$, where $A = LU$ is the Gaussian decomposition of the matrix $A$. Although this method is inherently parallel, an efficient implementation of this method in a distributed computing environment is non-trivial because of large amounts of interprocessor communication. Depending on several parameters which must be supplied, the amount of computer memory for calculating the projective approximate inverses can be estimated in advance.

  As in the symmetric case, for this projection method a variant utilizing the pattern of the original coefficient matrix as fixed prescribed projection pattern, called `FSAI`, is known. The `FSAI` algorithm requires no parameters, and is thus easy to use, but, on the other hand, cannot be tuned in case of unsatisfactory convergence of the preconditioned iteration. Both the set-up time and the iterative solve can efficiently be done in parallel.

The above summary of algorithmic properties of the preconditioning techniques indicates that a comparison of these methods is complicated. All these preconditioning methods differ in the amount of fill-in in the preconditioner, and hence in computer memory required, in the resulting acceleration of the iterative solver, and in the potential for parallelization. Therefore, we make the following selections for our test environment:
For the methods that produce a one-sided preconditioner (`SPAI` and `Plain projection`), we precondition the original linear system from the right-hand side; for all other methods, two-sided preconditioning is used. For all methods with adjustable

amount of fill-in in the preconditioner ($\mathtt{L^TL}$`-projection`, `AINV`, `SPAI`, `Plain projection` and `LU-projection`), we try to select the corresponding parameters in such a way that the resulting number of non-zeros in the obtained preconditioner is comparable.

We compare the new preconditioning techniques, i.e. the $\mathtt{L^TL}$`-projection`, `LU-projection` and `Plain projection`, to some of the state-of-the-art preconditioning approaches, by comparing the acceleration of the iterative solution process, the accuracy of the obtained approximate solution of the linear system, and by comparing the computational cost (i.e. by the number of floating-point operations) for forming the preconditioners and for solving the preconditioned linear systems. Note that comparing the number of floating-point operations for the different methods gives only limited evidence on the performance of these methods, since optimization aspects, like the particular potential for vectorization and parallelization, or cache reuse, are not taken into consideration.

For simplicity, we do all tests on a one-processor machine. Further, for comparability, we exclude the iterative nature of projection methods from the numerical tests, i.e. no preconditioned iteration is interrupted for improving the current projective approximate inverse by allowing more fill-in in case of unsatisfactory convergence (see the remarks given on pages 66 and 73 on this matter).

Although our selection of preconditioning techniques for the numerical tests does not cover all known approaches, it is suitable for considering the potential of the new projection methods ($\mathtt{L^TL}$`-projection`, `Plain projection` and `LU-projection`). The implicit methods (`IC(0)` and `ILU(0)`) are – although more sophisticated variants of these approaches exist – known to be quite efficient and robust. The efficiency of the recently proposed explicit techniques, `AINV` and `SPAI`, is roughly comparable to the efficiency of the implicit methods (see e.g. [9], [11]). Importantly, the explicit methods have much more potential for parallelization than the implicit methods.

## 7.1 Implementation Details

We implemented the projection methods (i.e. $\mathtt{L^TL}$`-projection`, `fixed-LU-projection`, `SPAI`, `Plain projection` and `LU-projection`) and the incomplete decompositions (i.e. `IC(0)` and `ILU(0)`) with the Fortran-90 language and we represented all floating-point numbers in double precision. The Fortran-77 sources of the `AINV` method were kindly provided by M. Benzi and M. Tůma (the tests published in [11] were done using this code).
All tests were computed on an one-processor machine.

### 7.1.1 The Linear Solvers for the Preconditioned Linear Systems

In this section, we describe our implementation of the iterative linear solvers considered in our numerical tests.

**Stopping Criteria**

The iterative linear solvers considered in our numerical tests are `CG` (see 3.5.1), `PRES20` (see 3.5.2), `BiCGstab` (see 3.5.3) and `ATPRES` (see 3.5.4). The exact solution $x \in I\!R^n$ of the considered linear systems is a random vector with $-1 \leq | x_i | \leq 1$ for $i = 1, \ldots, n$, and the right-hand side vector $b \in I\!R^n$ is set to $b := A \cdot x$. Thus, the errors $\left\| x - x_k \right\|_2$, where $x_k$ denotes the $k$-iterate of the induced iteration of the original linear system (according to definition 3.1), are known for all tests.
The stopping criterion for the iterative solvers is a relative decrease of twelve orders of magnitude of the residuals of the preconditioned iteration, i.e. the iterative solver has converged, if $\left\| \bar{r}_k \right\|_2 < 10^{-12} \cdot \left\| \bar{r}_0 \right\|$, where $\bar{r}_k$ denotes the $k$-th residual of the preconditioned iteration, formally defined by $\bar{r}_k := P_L A P_R y_k - P_L b$, with $y_k$ denoting the $k$-th iterate of the preconditioned iteration. If this stopping criterion is not attained after 1000 iterations, we consider the iterative solver to have diverged.

**Observation of Round-Off Errors in the Iteration**

The iterative solvers considered in our numerical tests belong to the family of conjugate Krylov subspace methods (see section 3.5). The residuals $\bar{r}_k$ of the solver iterations are calculated by updating the previous residual, i.e. according to

$$\bar{r}_k = \sum_{i=1}^{k} \beta_{i,k} \left( P_L A P_R \right)^i \bar{r}_0 + \bar{r}_0. \tag{7.1}$$

For growing $k$, the amount of accumulated round-off errors in the updated residuals becomes larger due to the finite precision arithmetic of computers. Particularly, if the considered coefficient matrix is ill-conditioned, the influence of these round-off errors can become inacceptably large. Thus, in our numerical tests, we safeguard the iterative solution process of the preconditioned linear systems by monitoring the relative deviation of the updated preconditioned residuals $\bar{r}_k$ from (7.1), as calculated by the particular iterative solvers, from the "exact" residuals $\bar{r}_k := P_L A P_R y_k - P_L b$, i.e. we monitor the size of

$$\frac{\left\| \bar{r}_k \right\|_2 - \left\| P_L A P_R y_k - P_L b \right\|_2}{\left\| P_L A P_R y_k - P_L b \right\|_2}, \tag{7.2}$$

where $y_k$ denotes the $k$-th iterate of the preconditioned iteration. During all our numerical tests, this relative deviation is only a few times larger than $10^{-5}$. Since a relative deviation of the updated residuals from the exact residuals larger than

$10^{-5}$ indicates numerical instabilities due to roundoff-errors in finite precision, the iteration is considered to have diverged in these cases and thus is stopped. These cases are designated in the corresponding sections.

**Normalization**

In order to promote numerical stability, all linear systems considered in our numerical tests were subjected to some diagonal scaling before applying any preconditioning technique or linear solver (see section 6.1 and the references therein for more information on the effect of diagonal scaling).

Before applying any iterative solver or preconditioning technique, the symmetric linear systems $Ax = b$ are scaled from both sides with the diagonal matrix

$$D_S := \begin{pmatrix} \left(\sum_{j=1}^{n} |a_{1j}|\right)^{-\frac{1}{2}} & & \\ & \ddots & \\ & & \left(\sum_{j=1}^{n} |a_{nj}|\right)^{-\frac{1}{2}} \end{pmatrix}, \tag{7.3}$$

where $a_{lk}$ denotes the $(l, k)$-th entry of the coefficient matrix $A$. Hence, instead of considering the original linear system $Ax = b$, the normalized linear system

$$D_S A D_S x = D_S b \tag{7.4}$$

is considered, with the diagonal matrix $D_S$ defined in (7.3).

Analogously, before applying any iterative solver or preconditioning technique to the unsymmetric linear systems, the linear system is scaled from the left-hand side with the diagonal matrix

$$D_U := \begin{pmatrix} \operatorname{sign}(a_{11}) / \sum_{j=1}^{n} |a_{1j}| & & \\ & \ddots & \\ & & \operatorname{sign}(a_{nn}) / \sum_{j=1}^{n} |a_{nj}| \end{pmatrix}, \tag{7.5}$$

where $a_{kk}$ denotes the $k$-th diagonal entry and $\operatorname{sign}(a_{kk})$ denotes the signum of the $k$-th diagonal entry of the coefficient matrix $A$. After that, the columns of the unsymmetric linear systems are scaled in such a way that the largest element in each column of the obtained coefficient matrix has an absolute value of one. Hence, instead of considering the original non-symmetric linear system $Ax = b$, the normalized linear system

$$D_U A C y = D_U b, \; Cy = x, \tag{7.6}$$

is considered, with the diagonal matrix $D_U$ defined in (7.5) and the diagonal matrix $C$ defined by

$$C := \begin{pmatrix} \left(\max_{1\le i\le n} |\hat{a}_{i,1}|\right)^{-1} & & \\ & \ddots & \\ & & \left(\max_{1\le i\le n} |\hat{a}_{i,n}|\right)^{-1} \end{pmatrix},$$

where $\hat{a}_{i,j}$ denotes the $(i,j)$-th element of the scaled coefficient matrix $D_U A$.

### 7.1.2 The Implementation of the Projection Methods

In this section, we discuss our implementation of the projection methods considered in our numerical tests, which are the `SPAI` algorithm (algorithm 12), the `Plain projection` algorithm (algorithm 13), the `L`$^{\mathtt{T}}$`L-projection` algorithm (algorithm 15) and the `LU-projection` algorithm (algorithm 16).
From the remarks given to the general pattern-adaptive projection methods (see chapter 5), as well as the remarks given to the particular algorithms (`SPAI`, `Plain projection`, `L`$^{\mathtt{T}}$`L-projection` and `LU-projection`), it is obvious, that numerous variants for the practical implementation are possible.

**Numerical Dropping in the Pattern Derivation Process**
In the pattern derivation process for the projection methods, numerical dropping can be applied with the aim of saving CPU-time. The fewer non-zeros are contained in the $J_k$-residual, the cheaper both the calculation of the corresponding residual-minimizing $J_k$-residuals $s_{J_k}$ and the computation of the decrease rates may be. On the other hand, the quality of the resulting projective approximate inverse may deteriorate.
Since we are interested in giving a general overview of the properties of the new projection methods, i.e. `Plain projection`, `L`$^{\mathtt{T}}$`L-projection` and `LU-projection`, we dispense with any kind of numerical dropping in the pattern derivation process in our implementation of these methods. But we found it necessary to do this kind of numerical dropping in our implementation of the `SPAI` algorithm. Since the `SPAI` method is the computationally most expensive method, we set each entry of the $J_k$-residuals with absolute value less than $10^{-4}$ to zero. Although the computational cost of the `SPAI` method is reduced enormously by this numerical dropping, `SPAI` still remains the most expensive among all methods tested.

**Limiting the Number of Decrease Rates**
The number of decrease rates calculated along the pattern derivation may be limited, and, based on some heuristic, only a specified number of decrease rates for the pattern derivation may be calculated instead. Obviously, such a strategy is two-edged: while it may be useful for reducing the computational cost of the pattern

derivation, it may deteriorate the quality of the obtained preconditioner. Since we are interested in both exploring the behavior of the new projection methods (`Plain projection`, `L`$^{\mathtt{T}}$`L-projection`, `LU-projection`), and in comparing these methods with the known `SPAI` method, we refrained from limiting the number of decrease rates for all projection methods tested.

**The Linear Solver for the Inner Linear Systems**

Since for every new column pattern in the pattern derivation process of the projection methods considered in our numerical tests the corresponding column of the projective approximate inverse is calculated by solving a small linear system, in the following referred to by "**inner linear system**", the choice of the applied linear solver – the "**inner linear solver**" – is important, both for the overall computational cost of the pattern derivation and for the quality of the obtained preconditioner.

For the `Plain projection` method, for the `L`$^{\mathtt{T}}$`L-projection` method and for the `LU-projection` method, the inner linear systems have coefficient matrices of the form $A(J_k, J_k)$ (or $(A(J_k, J_k))^T$), where $J_k$ denotes some column pattern generated in the pattern derivation process. In general applications, it is impossible to guarantee the non-singularity of this $J_k$-reduced matrices $A(J_k, J_k)$ (or $(A(J_k, J_k))^T$) a priori (which is theoretically demanded by lemma 4.5). Along our numerical tests, the situation of singular $A(J_k, J_k)$ never occurred. Still, these inner linear systems possibly are nearly singular and ill-conditioned.

In our implementation of the `L`$^{\mathtt{T}}$`L-projection` method, we utilized the `CG` iteration with the normalization from equation (7.4) for solving the inner linear systems. We limited the number of iterations allowed for the inner linear solver to at most equal the dimension of the inner linear system. Further, we interrupted the inner iteration, if the corresponding residual was less than $10^{-5}$.

In our implementation of the projection methods for the non-symmetric linear systems (i.e. `Plain projection`, `LU-projection` and `SPAI`), we found it necessary to utilize a dense $QR$-decomposition for solving the inner linear systems. The advantage of a dense $QR$-decomposition as inner linear solver is its reliability: if the considered inner linear system is numerically non-singular, it is solved with high accuracy. Further, a once calculated $QR$-decomposition can be updated with moderate computational cost, if a row or a column is added to the considered linear system (see e.g. [18] for details on this matter). Note that this situation occurs notoriously along the pattern derivation of the projection methods: every time the current column-pattern is augmented, a correspondingly augmented linear system must be solved.

We implemented iterative inner solvers (`PRES20`, `BiCGstab` and `ATPRES`) for the `Plain projection` method and the `LU-projection` method as well, but we found that for some problems of our test collection the quality of the obtained projective

approximate inverse was much worse, compared to the corresponding projective approximate inverse obtained with the $QR$-decomposition as inner solver.
Since the $QR$-decomposition turned out to be computationally much more expensive than the iterative methods, it is worthwhile to consider a cascaded approach: the reliability, and, too, the accuracy of iterative inner solvers may be promoted by applying some kind of preconditioning to them. For instance, some elaborated variant of `ILU` might be applied. Such an approach does not downgrade the parallelism of the considered projection methods, since the inner linear systems are treated on local processors.
Since we are interested in giving a general overview of the performance of `SPAI`, `Plain projection` and `LU-projection`, we confine ourselves to consider only the dense $QR$-factorization for solving the inner linear systems.

**The Decrease Rates for the Pattern Adaption**
For the projection methods with adaptive pattern derivation ($\mathtt{L^TL}$`-projection`, `Plain projection`, `LU-projection` and `SPAI`), we always use the diagonal pattern as the initial projection pattern, i.e. we define the initial projection pattern by $J_k^0 := (k)$ for $k = 1, \ldots, n$. Since none of the matrices considered in our numerical tests has a zero on the main diagonal, this choice guarantees, that all considered pattern adapting projection methods are explicit according to definition 4.3 in the beginning of the pattern derivation. As discussed in the following, the explicitness for the $\mathtt{L^TL}$`-projection` method, for the `Plain projection` method and for the `LU-projection` method is not guaranteed in later steps of the pattern derivation:
The theoretical background of the $\mathtt{L^TL}$`-projection` algorithm (as deduced in section 6.7.1) considers Hermitian positive definite matrices only. In our numerical tests we apply the $\mathtt{L^TL}$`-projection` algorithm to symmetric matrices which are not necessarily positive definite. Thus, if the considered matrix is symmetric but not positive definite, the $\mathtt{L^TL}$`-projection` algorithm is possibly not well defined. In particular, the decrease rates $\theta_j$ and $\lambda_j$, calculated in step (1(b)iv) of the formulation of the $\mathtt{L^TL}$`-projection` algorithm, given on page 130, are possibly not well-defined, or are possibly negative, if the considered matrix $A$ is not symmetric positive definite. Further, in this case, the $(J_k, J_k)$-reduced matrix $A\,(J_k, J_k)$, considered in steps (1a) and (1c) of the $\mathtt{L^TL}$`-projection` algorithm, may be singular.
Analogously, we apply the `Plain projection` algorithm and the `LU-projection` algorithm to non-symmetric matrices which are not necessarily positive real, as demanded by the theoretical background for these algorithms. Thus, the decrease rates may be not well defined, or negative, for these algorithms. Further, the inner linear systems, which have coefficient matrices of the form $A\,(J_k, J_k)$ (or $(A\,(J_k, J_k))^T$), may be singular.
However, during our extensive numerical tests for the $\mathtt{L^TL}$`-projection` algorithm, the `LU-projection` algorithm and for the `Plain projection` algorithm, no singular

inner linear system occurred at any time.
The case of negative decrease rates occurred in several tests. Note that, if the matrix $A \in I\!\!K^{n\times n}$ is indefinite, the expression $\|x\|_A^2$ is not a norm in $I\!\!K^n$, but the mnemonic abbreviation of $x^H Ax$. As a consequence, the decrease rates, calculated by the `L`$^{\mathtt{T}}$`L-projection` algorithm, by the `LU-projection` algorithm, and by the `Plain projection` algorithm may be negative. Thus, for choosing the new indices in the pattern derivation process it is possible to refer either to the actual values or to the absolute values of the decrease rates.
Our numerical tests done in the course of this work indicate, that for the pattern derivation process always the absolute values of the applied decrease rates should be applied.
Thus, in the numerical results stated for the `L`$^{\mathtt{T}}$`L-projection` algorithm, for the `LU-projection` algorithm, and for the `Plain projection` algorithm, we always consider the absolute values of the corresponding decrease rates.

## 7.2 The Test-Problems

In this section, we explain the generation of the test matrices from models of 3–dimensional Navier-Stokes and 2-dimensional Laplacian differential equations. Further, we briefly comment on the test problems taken from the MatrixMarket[4] collections (see [19]).

**The Navier–Stokes Model Based Problems**
Let

$$\bigtriangledown := \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)^T$$

and

$$\triangle := \bigtriangledown^T \bigtriangledown := \frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 y} + \frac{\partial^2}{\partial^2 z}.$$

We consider the solution of the partial differential equation

$$\triangle v + v + \rho \left(v^T \bigtriangledown\right) v = h \tag{7.7}$$

for the velocity

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

[4]These matrices can be downloaded from http://math.nist.gov/MatrixMarket.

with Dirichlet boundary conditions on the unit cube. The parameter $\rho$ in the partial differential equation (7.7) simulates a Reynolds number. The right-hand sides

$$h = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}$$

are determined in such a way that the partial differential equation (7.7) has the exact solution

$$\bar{v} = \begin{pmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \end{pmatrix} = \begin{pmatrix} \sin(2\pi x) & \cos(2\pi y) & \cos(2\pi z) \\ \cos(2\pi x) & \sin(2\pi y) & \cos(2\pi z) \\ \cos(2\pi x) & \cos(2\pi y) & \sin(2\pi z) \end{pmatrix}.$$

The linear systems arising from the partial differential equation (7.7) considered in our numerical tests are obtained from a finite difference discretization with consistency order 4 and from linearization with the first Newton step. The linear systems in our tests were generated with the FIDISOL program package [69]. The starting solution for the Newton method are the boundary conditions, which are 1-D interpolated on the whole domain [64]. The matrices were generated on a $20 \times 20 \times 20$ grid for the simulated Reynolds numbers $\rho = 1,\ 500,\ 10000$. The resulting matrices have 29 non-zero diagonals and the dimension $(24000 \times 24000)$.
The degree of difficulty for the matrices arising from the discretization of the partial differential equation (7.7) depends on the simulated Reynolds number $\rho$: the larger this number is, the more difficult the iterative solution of the corresponding linear system is. Circumstantial numerical tests with numerous different iterative methods are presented in [80]. In [79], the eigenvalues of such matrices on a coarser grid were computed. For small simulated Reynolds numbers $\rho$, the eigenvalues of the matrices lie in the positive halfplane close to the real axis. For larger $\rho$, the imaginary parts of eigenvalues become larger as well. For $\rho$ larger than some threshold value, the eigenvalues are scattered all across the complex plane.
Table 2 lists the matrices of this origin considered in our numerical tests. The dimension of the matrices in table 2 is denoted by $n$.

| matrix | $n$ | non-zeros | Reynolds number |
|---|---|---|---|
| c1m1o4 | 24000 | 274776 | 1 |
| c500m1o4 | 24000 | 274776 | 500 |
| c100000m1o4 | 24000 | 274776 | 100000 |

Table 2: Test problems from the Navier–Stokes model

Table 3 summarizes the results of the unpreconditioned iterative solvers applied to the linear systems (note that these are normalized, see equation (7.6)) with the coefficient matrices from table 2. We abbreviate by "its" the number of iterations of the solvers; a "$\oslash$" indicates that convergence was not attained in 1000 iterations. For the `BiCGstab` iteration, we additionally give the Euclidian norm of the final error, i.e. $\left\|x^* - x_k\right\|_2$, where $x^*$ denotes the exact solution of the linear system, and $x_k$ denotes the final approximation to $x^*$.

| matrix | `PRES20` | `BiCGstab` | | `ATPRES` |
|---|---|---|---|---|
| | its | its | $\|\|x^* - x_k\|\|_2$ | its |
| c1m1o4 | 193 | 72 | $3.9 \cdot 10^{-9}$ | 707 |
| c500m1o4 | $\oslash$ | $\oslash$ | | $\oslash$ |
| c100000m1o4 | $\oslash$ | $\oslash$ | | $\oslash$ |

Table 3: The unpreconditioned iterative methods applied to the matrices from the Navier–Stokes model

Note that the number of 1000 allowed iterations for a linear system with a dimension of 24000 is quite small. We remark, that, if considerably more iterations are allowed, the `BiCGstab` and the `ATPRES` iterations are known to converge, even for large Reynolds numbers.

**The Laplacian Model Based Problems**

We are interested in the solution of a boundary value problem on the unit square $G = (0,1) \times (0,1)$, i. e. in approximating the interior points of an equidistant square $(g+2) \times (g+2)$ grid. We use finite difference discretization with 5-point difference stars. Let the boundary value problem be

$$\begin{aligned} -\triangle u + \gamma\,(u_x + u_y) &= 0 && \text{for } (x,y) \in G \\ u\,(x,y) &= f\,(x,y) && \text{for } (x,y) \in \partial G, \end{aligned} \tag{7.8}$$

where $\triangle$ denotes the Laplacian operator, $\gamma \in I\!R$ is constant, and the function

$$f : I\!R \times I\!R \mapsto I\!R$$

is continuous on $\partial G$.

The matrix $A$ resulting from such a discretization is structural symmetric and has the block-tridiagonal form

$$A = \begin{pmatrix} D & U & & & \\ L & D & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & U \\ & & & L & D \end{pmatrix} \in I\!R^{g^2 \times g^2}. \tag{7.9}$$

The block matrices $L$, $U \in I\!R^{g\times g}$ are the diagonal matrices

$$L = \left(-1 - \frac{\gamma}{g+1}\right) I,$$
$$U = \left(-1 + \frac{\gamma}{g+1}\right) I,$$

where $I$ denotes the identity in $I\!R^{g\times g}$. The block matrices $D \in I\!R^{g\times g}$ have the tridiagonal form

$$D = tridiag\left(-1 - \frac{\gamma}{g+1}, 4, -1 + \frac{\gamma}{g+1}\right).$$

The parameter $\gamma$ from the partial differential equation (7.8) is a measure for the skew-symmetric part $\frac{1}{2}\left(A - A^T\right)$ of the matrix $A$ from (7.9), since

$$\frac{1}{2}\left(A - A^T\right) = \begin{pmatrix} \bar{D} & I & & & \\ -I & \bar{D} & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & I \\ & & & -I & \bar{D} \end{pmatrix} \cdot \frac{\gamma}{g+1},$$

where

$$\bar{D} := tridiag\,(-1, 0, 1)\,.$$

The symmetric part $\frac{1}{2}\left(A + A^T\right)$ of $A$ is independent of the parameter $\gamma$.
For our numerical tests, we set $g := 50$ and we consider the parameters $\gamma = 1$, 100 and 1000. Table 4 summarizes the matrices of this kind which we considered in our numerical tests, where $n$ denotes the dimension of the matrices.

| matrix | $n$ | non-zeros | $\gamma$ |
|---|---|---|---|
| l_50_1 | 2500 | 12300 | 1 |
| l_50_100 | 2500 | 12300 | 100 |
| l_50_1000 | 2500 | 12300 | 1000 |

Table 4: Test problems from the Laplacian model

In table 5, we summarize the results of the unpreconditioned iterative solvers applied to the linear systems with the coefficient matrices from table 4 (those linear systems were normalized before applying the iterative methods, see equation (7.6)). As before, by "its" we abbreviate the number of iterations until convergence; a "$\oslash$"

indicates that the iteration did not converge in 1000 iterations. For `BiCGstab`, we give the the Euclidian norm of the final error, i.e. of the difference between the approximate and the exact solution of the considered linear system.

| matrix | `PRES20` | `BiCGstab` | | `ATPRES` |
|---|---|---|---|---|
| | its | its | $\|\|x^* - x_k\|\|_2$ | its |
| l_50_1 | 496 | 118 | $5.2 \cdot 10^{-9}$ | $\oslash$ |
| l_50_100 | 266 | 215 | $5.8 \cdot 10^{-11}$ | 424 |
| l_50_1000 | 531 | $\oslash$ | | 683 |

Table 5: The unpreconditioned iterative methods applied to the matrices from the Laplacian model

**The Problems from MatrixMarket Collections**

The example problems from the MatrixMarket collections (see [19]) arise from a wide range of different practical applications. The degree of difficulty of the chosen test problems varies between simple (1138bus, orsirr2) and substantial (s3rmq4m1, utm3060). The largest symmetric matrices (s1rmq4m1, s3rmq4m1) have a dimension of 5489 and contain 143300 non-zeros in their upper triangular part. The largest unsymmetric matrix (memplus) has a dimension of 17758 and contains 126150 non-zeros. Tables 6 and 8 summarize the symmetric and the non-symmetric matrices considered in our tests, where the dimension of the matrices is abbreviated by $n$.

| matrix | $n$ | non-zeros | Application | Origin |
|---|---|---|---|---|
| 1138bus | 1138 | 2596 | Power system admittance | Harwell–Boeing |
| nasa2910 | 2910 | 88603 | Structural analysis | Tim Davis Collection |
| bcsstk21 | 3600 | 15100 | Structural engineering | Harwell–Boeing |
| bcsstk23 | 3134 | 24156 | Structural engineering | Harwell–Boeing |
| s1rmq4m1 | 5489 | 143300 | Structural mechanics | Cylshell |
| s3rmq4m1 | 5489 | 143300 | Structural mechanics | Cylshell |

Table 6: Symmetric test problems from the MatrixMarket collections

In table 7, we present the results obtained by the unpreconditioned `CG` method applied to the symmetric linear systems from the MatrixMarket test problems (note that the linear systems are normalized according to (7.4)). The symbol "$\oslash$" indicates that the iteration did not converge within 1000 steps. For the converging cases, we give the final error as well.

| matrix | CG | |
|---|---|---|
| | its | $\|\|x^* - x_k\|\|_2$ |
| 1138bus | $\oslash$ | |
| nasa2910 | $\oslash$ | |
| bcsstk21 | 660 | $1.4 \cdot 10^{-3}$ |
| bcsstk23 | $\oslash$ | |
| s1rmq4m1 | 846 | $9.0 \cdot 10^{-5}$ |
| s3rmq4m1 | $\oslash$ | |

Table 7: Performance of the `CG` method applied to the symmetric MatrixMarket problems

| matrix | n | nz | Application | Origin |
|---|---|---|---|---|
| orsirr2 | 886 | 5970 | Oil reservoir simulation | Harwell–Boeing |
| pores2 | 1224 | 9613 | Reservoir modeling | Harwell–Boeing |
| sherman2 | 1080 | 23094 | Oil reservoir modeling | Harwell–Boeing |
| saylr4 | 3564 | 22316 | Oil reservoir modeling | Harwell–Boeing |
| memplus | 17758 | 126150 | Memory circuit design | Hamm |
| utm1070b | 1700 | 21509 | Nuclear physics (plasmas) | Sparskit |
| utm3060 | 3060 | 42211 | Nuclear physics (plasmas) | Sparskit |

Table 8: The unsymmetric test problems from the MatrixMarket collections

Table 9 summarizes the results obtained with the unpreconditioned iterations for the non-symmetric problems (normalized according to (7.6)) from the MatrixMarket collections. The "$\oslash$" indicates, that the iteration did not converge within 1000 iterations. For the `BiCGstab` iteration we additionally give the final error for the converging cases.

| matrix | PRES20 | BiCGstab | | ATPRES |
|---|---|---|---|---|
| | its | its | $\|\|x^* - x_k\|\|_2$ | its |
| orsirr2 | 668 | 367 | $1.1 \cdot 10^{-7}$ | $\oslash$ |
| pores2 | $\oslash$ | $\oslash$ | | $\oslash$ |
| sherman2 | $\oslash$ | $\oslash$ | | $\oslash$ |
| saylr4 | $\oslash$ | $\oslash$ | | $\oslash$ |
| memplus | $\oslash$ | 765 | $3.9 \cdot 10^{-6}$ | $\oslash$ |
| utm1070b | $\oslash$ | $\oslash$ | | $\oslash$ |
| utm3060 | $\oslash$ | $\oslash$ | | $\oslash$ |

Table 9: Performance of the unpreconditioned iterative solvers applied to the unsymmetric MatrixMarket problems

The results given in tables 3, 5, 7 and 9 make plain that the considered iterative methods perform unsatisfactorily for the problems in our test collection. In the remainder of this chapter, we consider how the performance of the iterative solvers is promoted by different kinds of preconditioners.

## 7.3 An Illustrated Example

In this section, we demonstrate the potential of pattern adaptive projection methods by considering the `Plain projection` method applied to the matrix orsirr2 (this matrix is contained in the Harwell–Boeing collection, see table 8 and [22]). In particular, we compare the pattern of the true inverse of the matrix orsirr2 to three different projective approximate inverses determined by the `Plain projection` method. Further, we compare the eigenvalue distribution of the original matrix orsirr2 and of the three preconditioned matrices. Finally, we examine the convergence of the `BiCGstab` solver applied to the unpreconditioned and to the three preconditioned linear systems.
The matrix orsirr2 is an element of $\mathbb{R}^{886\times 886}$, has 5970 non-zeros, and is non-symmetric (note that we actually consider the normalized matrix orsirr2, according to (7.6)).

First of all, we calculated the exact inverse of the matrix orsirr2 by calculating its $QR$-decomposition orsirr2 $= QR$, with $Q$ orthogonal and $R$ upper triangular, and then by computing the $k$-th column of the inverse by $R^{-1}Q^T u_k$ for $k = 1, \ldots, 886$, where $u_k$ denotes the $k$-th unit vector in $\mathbb{R}^{886}$. The obtained inverse of the matrix orsirr2 has 784996 non-zeros, i.e. is completely dense.
But, although the exact inverse of the matrix orsirr2 is completely dense, the absolute values of the entries in the inverse widely differ in their size. The largest entry in absolute value of the inverse is about 265; the average of all entries in absolute value is about 2.76. If all entries of the inverse with an absolute value less than 30 are set to zero, we obtain a matrix with 13821 non-zeros only. In the following, we denote this matrix by orsirr2_13821. Figure 2 shows a plot of the distribution of the non-zeros in the matrix orsirr2_13821.
The exact inverse of the matrix orsirr2 contains elements which widely differ in their size. Thus it is promising to approximate the inverse of orsirr2 with a sparse matrix, i.e. with about the same amount of non-zeros as contained in the matrix orsirr2, such that the approximate inverse is an efficient preconditioner.
We calculated two different projective approximate inverses of the matrix orsirr2 by the `Plain projection` algorithm. The parameters for these projective approximate inverses are chosen in such a way that one of the resulting projective approximate inverses contains at most the same number of non-zeros as the original matrix orsirr2. The resulting projective approximate inverse has 3009 non-zeros; we denote

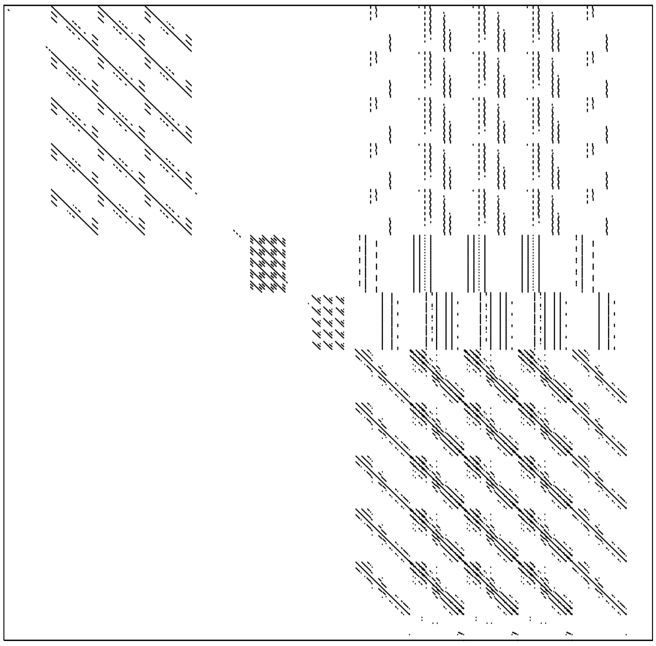

Figure 2: The pattern of the matrix orsirr2_13821. This is the exact inverse of orsirr2, where all entries with an absolute value less than 30 were set to zero. The matrix orsirr2_13821 has 13821 non-zeros.

this matrix by orsirr2_3009. The other projective approximate inverse was allowed to have about twice as many non-zeros as the matrix orsirr2_3009. The resulting projective approximate inverse contains 8060 non-zeros; we denote this matrix by orsirr2_8060.

Figures 3 and 4 show the patterns of projective approximate inverses orsirr2_3009 and orsirr2_8060. Note the similarity of the patterns of the projective approximate inverses orsirr2_3009 and orsirr2_8060 compared to the pattern of the dropped exact inverse orsirr2_13821. This similarity shows that the "large" entries of the exact inverse of the matrix orsirr2 are captured by the adaptive pattern derivation process of the `Plain projection` method. In [38], similar plots of approximate inverses of the matrix orsirr2, which are determined by the `SPAI` method, are shown.

Note that the non-singularity of the projective approximate inverses orsirr2_3009 and orsirr2_8060, shown in plots 3 and 4, is not guaranteed by theory. Further, no information on the distribution of the eigenvalues of the coefficient matrix of the preconditioned linear system, i.e. the matrix orsirr2 multiplied by orsirr2_3009 or orsirr2_8060, is available (see the brief discussion on pp. 32–34 and the references therein for comments on the connection between the eigenvalue distribution and the convergence of Krylov subspace methods). Hence, it is impossible to assess a priori whether or not those projective approximate inverses are efficient preconditioners for linear systems with the coefficient matrix orsirr2.

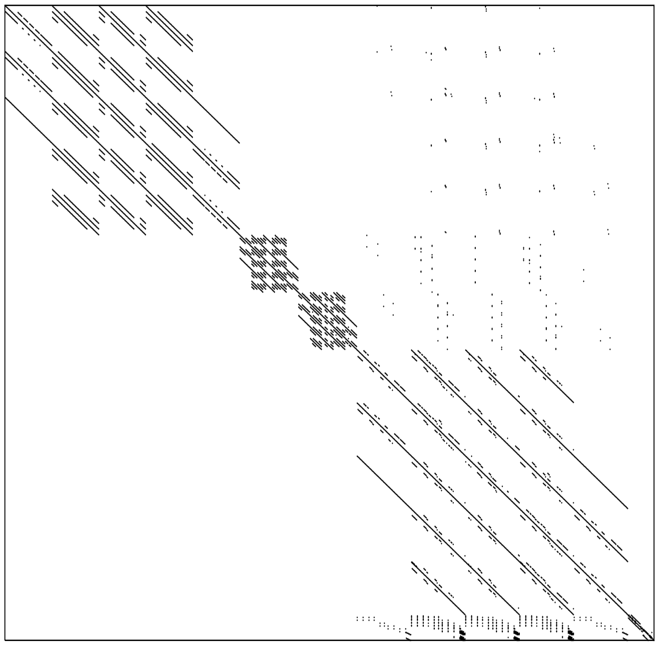

Figure 3: The pattern of the projective approximate inverse orsirr2_8060 of the matrix orsirr2, calculated by the `Plain projection` algorithm with 8060 non-zeros

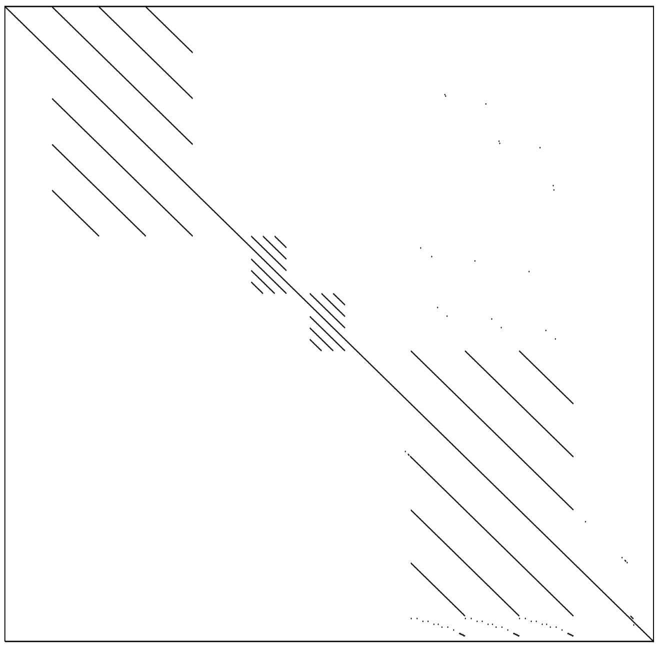

Figure 4: The pattern of the projective approximate inverse orsirr2_3009, calculated by the `Plain projection` algorithm with 3009 non-zeros

In order to obtain an approximate inverse of the matrix orsirr2, for which the non-singularity and the eigenvalue distribution of the preconditioned coefficient matrix is known by theory, we calculated a projective approximate inverse of orsirr2 by the `Plain projection` algorithm in such a way that the one-norm of each column of the preconditioned coefficient matrix is less than 0.9. In this situation, proposition 3.5 applies, stating that the preconditioned coefficient matrix is non-singular with eigenvalues clustered in a disk around one with a radius of 0.9. The obtained projective approximate inverse has 17031 non-zeros; we denote this matrix by orsirr2_17031. Note that the number of floating-point operations as well as the required computer memory for the calculation of the matrix orsirr2_17031 greatly exceeds the corresponding quantities required for calculating the projective approximate inverses orsirr2_3009 and orsirr2_8060.

The pattern of the projective approximate inverse orsirr2_17031 is shown in figure 5. The similarity of this plot compared to the plots of the matrices orsirr2_8060 and orsirr2_3009 (figures 3 and 4) suggests that the matrices orsirr2_8060 and orsirr2_3009 act on the eigenvalue distribution of the corresponding preconditioned linear systems, in a way comparable to the matrix orsirr2_17031.

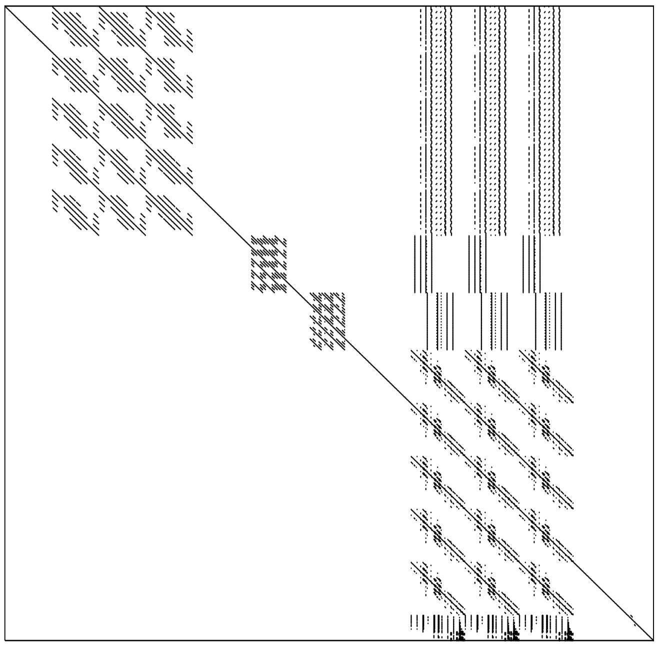

Figure 5: The pattern of the projective approximate inverse orsirr2_17031. The one-norm of each column of this matrix is less than 0.9

Figures 6, 7, 8 and 9 show the eigenvalue distribution of the original matrix orsirr2, and of the matrix orsirr2 multiplied by one of the three projective approximate inverses orsirr2_3009, orsirr2_8060 and orsirr2_17031.
Table 10 provides some insight in the distribution of the eigenvalues of the matrix orsirr2, and of the matrix orsirr2 multiplied by one of the three projective approximate inverses orsirr2_3009, orsirr2_8060 and orsirr2_17031. The left-hand side column of table 10 gives the radius of the considered disc around $(1, 0) \in \mathbb{C}$. The remainder of the corresponding lines in table 10 state the number of eigenvalues of the four particular matrices contained in these discs. Obviously, preconditioning the matrix orsirr2 with the matrices orsirr2_3009, orsirr2_8060 or orsirr2_17031 has the effect of clustering the eigenvalues around the point $(1, 0)$ in the complex plane.

| radius around $(1,0) \in \mathbb{C}$ | orsirr2 | orsirr2 $\cdot$ orsirr2_3009 | orsirr2 $\cdot$ orsirr2_8060 | orsirr2 $\cdot$ orsirr2_17031 |
|---|---|---|---|---|
| 0.1 | 174 | 448 | 450 | 439 |
| 0.2 | 182 | 632 | 638 | 638 |
| 0.3 | 186 | 694 | 750 | 764 |
| 0.4 | 194 | 764 | 810 | 827 |
| 0.5 | 198 | 828 | 846 | 856 |
| 0.6 | 202 | 838 | 863 | 870 |
| 0.7 | 318 | 849 | 870 | 881 |
| 0.8 | 546 | 860 | 880 | 886 |
| 0.9 | 546 | 876 | 884 | 886 |
| 1.0 | 886 | 886 | 886 | 886 |

Table 10: The number of eigenvalues in discs around $(1, 0) \in \mathbb{C}$ of orsirr2, and of orsirr2 multiplied by orsirr2_3009, orsirr2_8060 or orsirr2_17031

The eigenvalues of the original matrix orsirr2 are close to the real axis, loosely clustered in five spots, where one of them is close to the origin, with real parts between $3.9 \cdot 10^{-4}$ and 2. Only sixteen eigenvalues of the matrix orsirr2, shown in figure 6, have a non-vanishing imaginary part (the largest imaginary part has an absolute value of $2.2 \cdot 10^{-3}$).
The eigenvalues of the matrix orsirr2 multiplied by the projective approximate inverse orsirr2_3009 have real parts between $2.8 \cdot 10^{-2}$ and 1.67. The number of eigenvalues with non-vanishing imaginary parts is 472 (the largest imaginary part has an absolute value of 0.304). Compared to the eigenvalues of the original matrix orsirr2, the number of complex eigenvalues is drastically enlarged. The percentage of eigenvalues contained in the disc with radius 0.5 around $(1, 0) \in \mathbb{C}$ is 93.4. The corresponding percentage for the original matrix orsirr2 is only 22.3.

Similar observations – with even more clustering of the eigenvalues around $(1,0) \in \mathbb{C}$ – hold for the product matrices formed between orsirr2 and orsirr2_8060, or respectively orsirr2_17031.

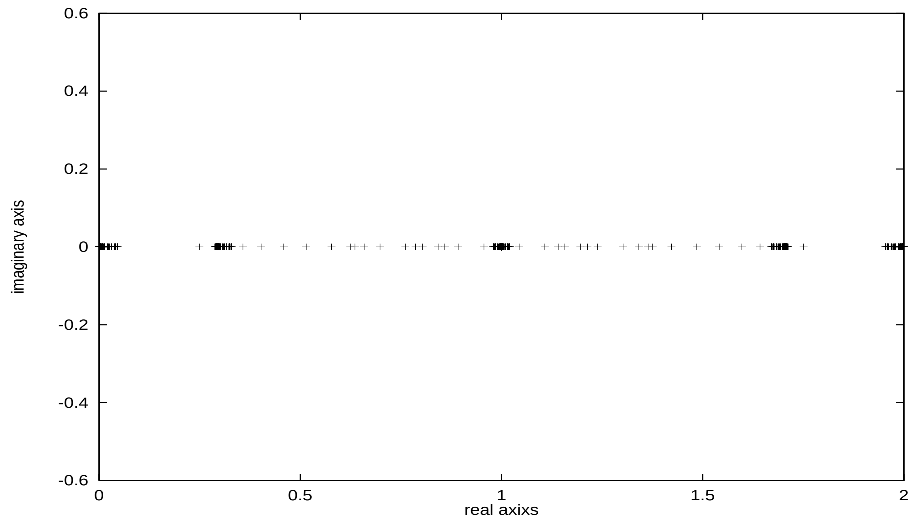


Figure 6: The eigenvalues of the matrix orsirr2

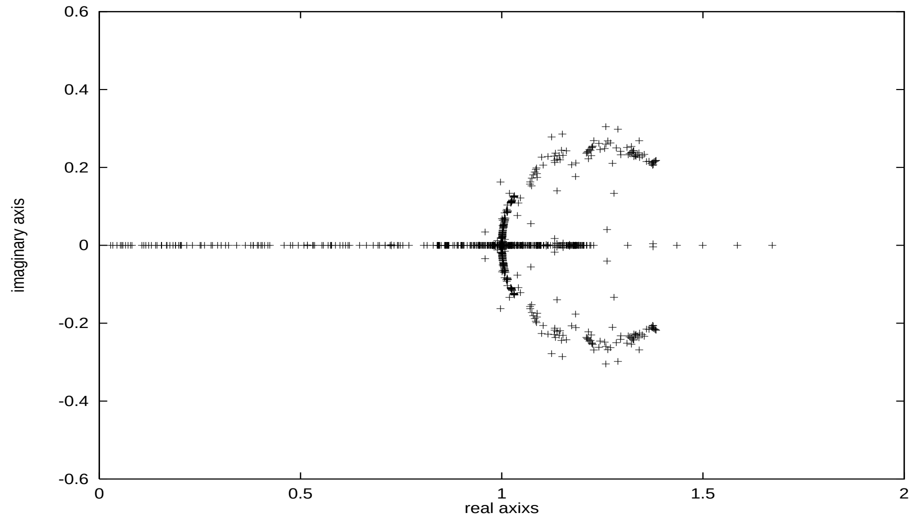


Figure 7: The eigenvalues of matrix orsirr2 multiplied by the projective approximate inverse orsirr2_3009

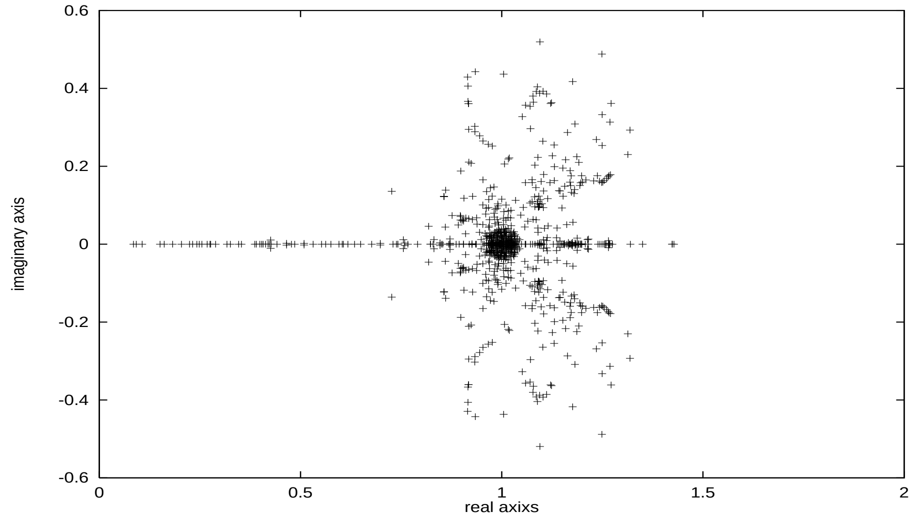


Figure 8: The eigenvalues of matrix orsirr2 multiplied by the projective approximate inverse orsirr2_8060

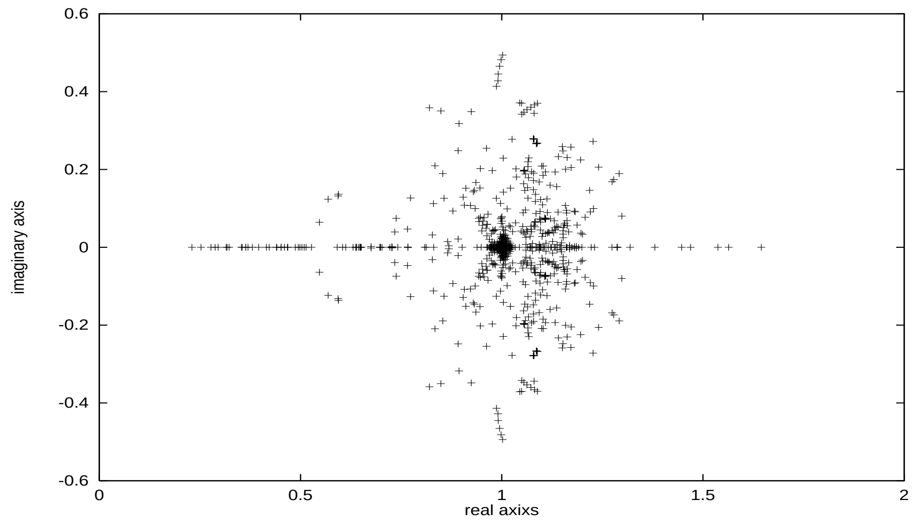


Figure 9: The eigenvalues of matrix orsirr2 multiplied by the projective approximate inverse orsirr2_17031

Finally, we consider the convergence of the `BiCGstab` solver applied to the original system with the coefficient matrix orsirr2, and to the linear systems preconditioned with the matrices orsirr2_3009, orsirr2_8060 and orsirr2_17031 (the design of those linear systems and the stopping criterion for the iteration is described in section 7.1.1).

Figure 10 portrays the performance of the four iterations. The unpreconditioned iteration is represented by the curve "no_prec". The curves "pp_3009", "pp_8060" and "pp_17031" denote the correspondingly preconditioned iterations.
The upper two graphs consider the residuals of the iterations. Note that, for the preconditioned iterations, the preconditioned residuals – and not the induced original residuals (see definition 3.1) – are plotted. On the vertical axes of the two upper graphs, the decadic logarithm of the relative residuals (i.e. $\log\left(\frac{||r_k||_2}{||r_0||_2}\right)$, where $r_k$ denotes the $k$-th residual of the iteration for $k \geq 0$) is settled. The upper left-hand side graph compares the residual decrease to the number of iteration steps; the upper right-hand side graph compares the residuals to the number of floating-point operations.
On the vertical axis of the lower graph, the decadic logarithm of the relative errors (i.e. $\log\left(\frac{||e_k||_2}{||e_0||_2}\right)$, where $r_k$ denotes the $k$-th original error of the iteration for $k \geq 0$) is settled. Note that, for the preconditioned iterations, not the errors of the preconditioned iteration but the errors of the corresponding original iterations (see definition 3.1) are shown. The lower graph shows the behavior of the errors correlated to the iteration steps.
We dispense with plotting the curve "pp_17031" in the upper right-hand side graph, because the set-up cost for the preconditioner orsirr2_17031 is about $2.8 \cdot 10^9$ floating-point operations. Thus, if this curve is displayed in the upper right-hand side graph, the three other curves become indistinguishable from the vertical axis.
The unpreconditioned iteration requires 367 iteration steps. The number of iterations is drastically reduced by applying any of the three preconditioners orsirr2_3009 (51 iterations), orsirr2_8060 (36 iterations) or orsirr2_17031 (20 iterations). Considering the set-up cost for the three preconditioners, we see that the number of floating-point operations for the matrix orsirr2_17031 is prohibitively high. As seen in figure 10, the set-up cost for the matrices orsirr2_3009 and orsirr2_8060 is acceptable. Both preconditioned iterations have an overall computational cost smaller that the unpreconditioned iteration.
Altogether, the above considerations indicate that the `Plain projection` algorithm is a promising preconditioning technique. In particular, the `Plain projection` method has the potential to determine adaptively useful patterns of approximate inverses, in such a way that these are efficient accelerators for iterative solvers. Further, the eigenvalues of the preconditioned systems tend to be clustered around the point $(1, 0)$ in the complex plane.
In the remainder of this section, we survey the properties of the `Plain projection` method, and the $\mathtt{L^TL}$`-projection` and `LU-projection` methods, by considering numerous test problems and by comparing these methods to the state-of-the-art methods.

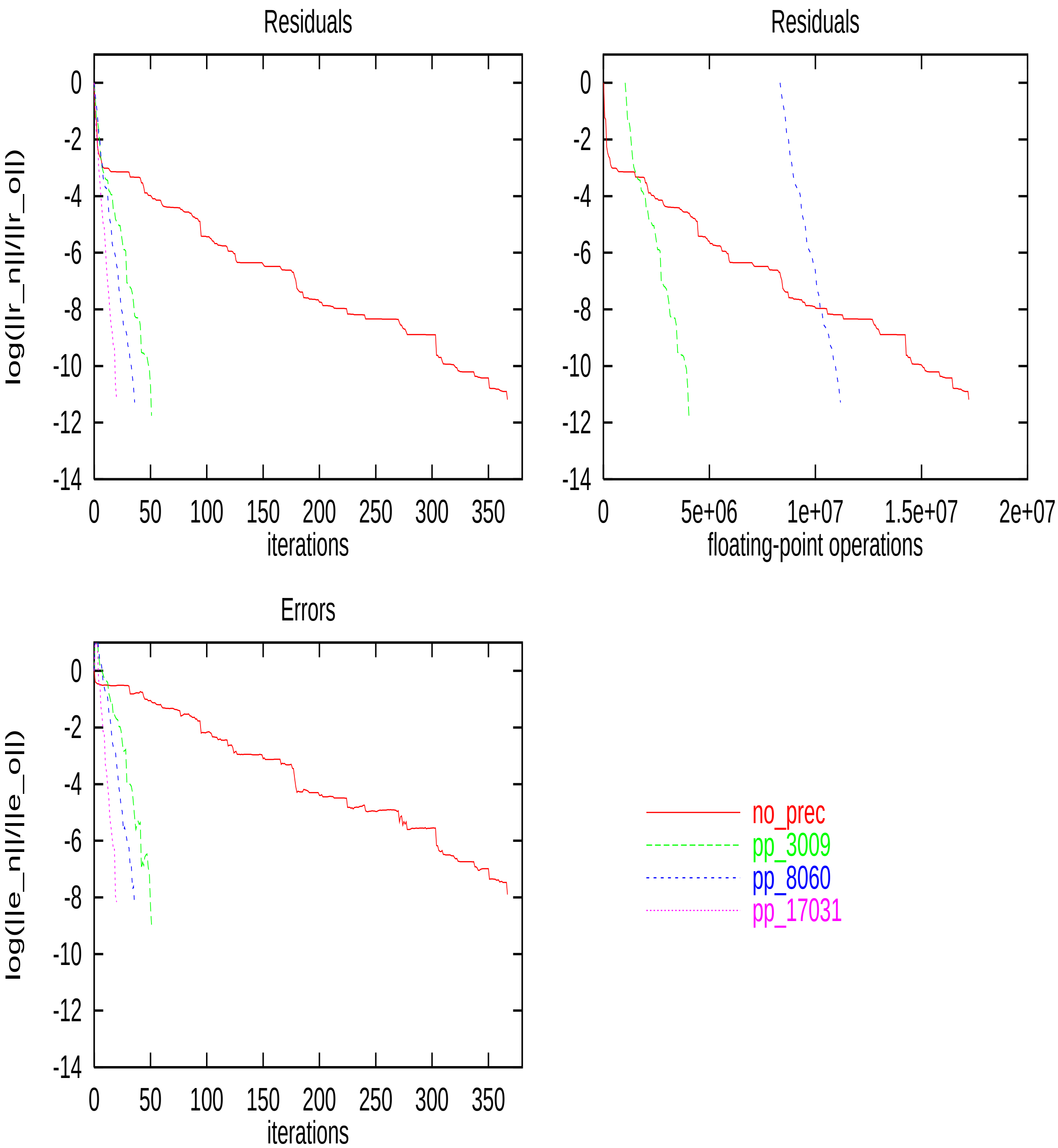


Figure 10: Convergence of `BiCGstab` applied to orsirr2 with and without preconditioning

## 7.4 Numerical Experiments for the Symmetric Matrices

In this section, we describe the performance of the $\mathtt{L^TL}$-projection algorithm (algorithm 15) applied to the symmetric matrices from our test set. Further, we compare the results obtained by the $\mathtt{L^TL}$-projection algorithm to the results obtained by the AINV algorithm (algorithm 14), by the IC(0) algorithm (see section 6.5) and by the FSAI algorithm (this algorithm coincides with the $\mathtt{L^TL}$-projection algorithm by defining the fixed projection pattern equal to the pattern of the original matrix). For all tests with the symmetric matrices, we consider the CG iterative solver.
In table 11, we summarize the results obtained by applying the CG solver without preconditioning and by applying the IC(0) method. We state both the number of iteration steps and, in case of convergence within 1000 steps, the Euclidian norm of the final error of the induced iteration for the original linear system (see definition 3.1). The symbol "$\oslash$" indicates that convergence was not attained in 1000 iterations. The "†"-symbol for the matrices nasa2910, bcsstk21 and bcsstk23 indicates that the IC(0) algorithm broke down due to negative diagonal entries in the incomplete factor $P_L \approx L$. This problem occurs because symmetric matrices nasa2910, bcsstk21 and bcsstk23 are not positive definite. As indicated by the ‡-symbol, the CG iteration with IC(0) preconditioning for the matrix s3rmq4m1 is interrupted after 180 iterations due to an excessive accumulation of round-off errors (see page 155). The unpreconditioned CG iteration fails to converge on four problems. The IC(0) preconditioner causes a drastical acceleration of the convergence rate for the matrices 1138bus and s1rmq4m1).

| matrix | CG | | CG | |
|---|---|---|---|---|
| | IC(0) | $\|\|x^* - x_k\|\|_2$ | unprec. | $\|\|x^* - x_k\|\|_2$ |
| 1138bus | 154 | $3.2 \cdot 10^{-7}$ | $\oslash$ | |
| nasa2910 | † | – | $\oslash$ | |
| bcsstk21 | † | – | 660 | $1.4 \cdot 10^{-3}$ |
| bcsstk23 | † | – | $\oslash$ | |
| s1rmq4m1 | 107 | $2.6 \cdot 10^{-6}$ | 846 | $9.0 \cdot 10^{-5}$ |
| s3rmq4m1 | ‡ | | $\oslash$ | |

Table 11: Performance of the CG solver applied to the symmetric linear systems preconditioned by the IC(0) method and unpreconditioned

In table 12, we summarize the performance of the AINV preconditioner. The column "thresh." contains the particular dropping threshold for the biconjugate vectors (see section 6.6). The number of non-zeros in the particular preconditioning matrices is abbreviated by "nz($P$)". The two columns on the right-hand side state the number of iterations, and, in case of convergence, the Euclidian norm of the final error for the original system. As indicated by the symbol "$\oslash$", for four problems no choice for the dropping threshold could be found in such a way that the preconditioned iteration converged within 1000 iterations. This is, although in these cases fill-in of about three times of non-zeros in the upper triangular part of the original matrix was allowed. For the two converging cases, the choice of the dropping thresholds was very sensitive with regards to changes of the amounts of fill-in and with regards to the obtained convergence rate, too.

| matrix | thresh. | nz($P$) | CG | $\|\|x^* - x_k\|\|_2$ |
|---|---|---|---|---|
| 1138bus | 0.15 | 3428 | 128 | $3.5 \cdot 10^{-7}$ |
| nasa2910 | | | $\oslash$ | |
| bcsstk21 | 0.184 | 21067 | 372 | $3.5 \cdot 10^{-4}$ |
| bcsstk23 | | | $\oslash$ | |
| s1rmq4m1 | | | $\oslash$ | |
| s3rmq4m1 | | | $\oslash$ | |

Table 12: Performance of the `CG` solver applied to the symmetric linear systems preconditioned by the `AINV` method

Table 13 summarizes the results obtained by the `L`$^{\mathtt{T}}$`L-projection` algorithm. For each of the test matrices three rows are given. The row with the label "fixed" contains the results for the `FSAI` algorithm. The rows with the "`u`" label and with the "`m`" label correspond to the `L`$^{\mathtt{T}}$`L-projection` algorithm applied with the univariate decrease rates $\tilde{\theta}_j$ from (6.50) and with the decrease rates $\tilde{\lambda}_j$ from (6.51) respectively. For both variants of the pattern adaptive `L`$^{\mathtt{T}}$`L-projection` algorithms, the initial projection pattern is always the diagonal pattern, i.e. the initial pattern is defined by $J_k^0 := (k)$ for $k = 1, \ldots, n$.
The column labeled "nz($P$)" states the number of non-zeros in the obtained preconditioners.
For the `L`$^{\mathtt{T}}$`L-projection` algorithm, three control parameters for the pattern derivation must be supplied (those are explained in detail for algorithm 8, see pages 68–69). These parameters are

i) $\underline{mf}$ : The maximum number of non-zeros in each column of the projective approximate inverse.

ii) $\underline{ms}$ : The maximum number of pattern updating steps for each column-pattern.

iii) $\underline{mfps}$ : The maximum number of indices added in one updating step to the column-pattern.

Since for the $\mathtt{L^TL}$-projection algorithm only the quantitative stopping criterion can be applied (see pages 67–68), controlling the amount of fill-in in the preconditioner is possible. We tried to select the above parameters in such a way that the obtained projective approximate inverses have at most twice as many non-zeros as the corresponding original matrices. The particular choices for the parameters $mf$, $ms$, $mfps$, as stated in table 13, are obtained after a few test runs of each test problem. The sensitivity of the $\mathtt{L^TL}$-projection algorithms to variations of these parameters, regarding quality of the obtained preconditioners, depends on the tests problems and is not known a priori.

The FSAI algorithm fails on two of test problems; the iteration with the matrix bcsstk21 fails to converge, and, as indicated by the ‡-symbol, the iteration with the matrix s3rmq4m1, is interuppted after 580 iterations due to an excessive accumulation of round-off errors (see page 155). The $\mathtt{L^TL}$-projection algorithms with adaptive pattern derivation fail on one of the test problems. The iteration with the matrix s3rmq4m1 is interrupted after 320 iterations (with the $\tilde{\theta}_j$ decrease rates), and after 340 iterations (with the $\tilde{\lambda}_j$ decrease rates), because of an excessive accumulation of round-off errors (see page 155). This problem could not be alleviated by varying the control parameters for the pattern derivation.
The initial residuals of the iterations with the matrix bcsstk23 is for the two $\mathtt{L^TL}$-projection algorithms approximately $8.4 \cdot 10^8$. Although the the preconditioned iterations converge, the final errors are large. The initial errors for both iterations are about $9.1 \cdot 10^8$. Hence, for obtaining a better approximation to the solution of the preconditioned linear systems with bcsstk23, more iteration steps, and hence a stricter stopping criterion for the iteration is necessary. However, this is a difficult task in a double-precision arithmetic. For problems like these, an arithmetic with a higher accurary should be utilized to prevent exessive accumulation of round-off errors. See e.g. [44] for a detailed survey on high precision computing.

In practice, iterative solvers for linear systems are controlled by monitoring the norms of the residuals, because the errors are unknown. The norms of the residuals and errors are connected as stated by (3.5) for an unpreconditioned iteration, and by (3.12) for a preconditioned iteration. If the coefficient matrix of the considered linear system has a small condition number, the connection between the residuals and errors is strong, i.e. a decrease of the norms of the residuals indicates a corresponding decrease of the error-norms. Conversely, if the considered coefficient matrix has a large condition number, the connection between the residuals and errors is weak, i.e. a decrease of the residual norms in the course of the iteration does not necessarily indicate correspondly decreasing error norms (such an example is the

| matrix | decrease rates | nz($P$) | $\mathtt{L^TL}$-projection | | | CG | |
|---|---|---|---|---|---|---|---|
| | | | $mf$ | $ms$ | $mfps$ | its | $\|\|x^* - x_k\|\|_2$ |
| 1138bus | fixed | 2596 | – | – | – | 228 | $4.9 \cdot 10^{-7}$ |
| | u | 2723 | 3 | 2 | 1 | 172 | $9.2 \cdot 10^{-7}$ |
| | m | 2723 | 3 | 2 | 1 | 172 | $8.4 \cdot 10^{-7}$ |
| nasa2910 | fixed | 88090 | – | – | – | 397 | $8.1 \cdot 10^{-5}$ |
| | u | 72314 | 26 | 5 | 5 | 190 | $1.0 \cdot 10^{-4}$ |
| | m | 72315 | 26 | 5 | 5 | 195 | $7.7 \cdot 10^{-5}$ |
| bcsstk21 | fixed | 15100 | – | – | – | 354 | $6.2 \cdot 10^{-4}$ |
| | u | 21523 | 6 | 5 | 1 | 290 | $4.3 \cdot 10^{-4}$ |
| | m | 21523 | 6 | 5 | 1 | 282 | $6.8 \cdot 10^{-4}$ |
| bcsstk23 | fixed | 24156 | – | – | – | ∅ | |
| | u | 46673 | 15 | 7 | 2 | 575 | $9.6 \cdot 10^{1}$ |
| | m | 68121 | 22 | 7 | 3 | 669 | $3.7 \cdot 10^{2}$ |
| s1rmq4m1 | fixed | 143300 | – | – | – | 297 | $1.5 \cdot 10^{-5}$ |
| | u | 180609 | 33 | 8 | 4 | 163 | $3.4 \cdot 10^{-6}$ |
| | m | 180609 | 33 | 8 | 4 | 165 | $3.7 \cdot 10^{-6}$ |
| s3rmq4m1 | fixed | 143300 | – | – | – | ‡ | |
| | u | 202427 | 37 | 9 | 4 | ‡ | |
| | m | 202427 | 37 | 9 | 4 | ‡ | |

Table 13: Performance of the CG method with $\mathtt{L^TL}$-projection preconditioning

matrix bcsstk23, see figure 14). Thus, for theoretical investigations, it is important to consider both residuals and errors. Therefore, and for obtaining an impression of the performance of the considered preconditioners, we give in figures 11–16 the plots obtained for the symmetric test problems. The curve with the label "unpreconditioned" is obtained for the unpreconditioned CG iteration. The curve with the label "FSAI" is obtained for the FSAI preconditioner. Curves "$\mathtt{L^TL}$-projection (u)" and "$\mathtt{L^TL}$-projection (m)" are obtained for the $\mathtt{L^TL}$-projection algorithm with the adaptive pattern derivation based on the univariate decrease rates $\tilde{\theta}_j$, and $\tilde{\lambda}_j$ respectively. The labels IC(0) and AINV designate the curves obtained for the corresponding preconditioners. The layout of the graphs is as explained on page 173.

**Comparison of the Preconditioners for Symmetric Linear Systems**

The unpreconditioned CG iteration has, in general, the slowest rate of convergence; it fails to converge in four out of six test problems. In the two converging tests, the unpreconditioned iteration is the slowest of all, both measured by iterations and by

floating-point operations.
Only for two test problems (1138bus and bcsstk21), the AINV algorithm produces efficient preconditioners. In the converging cases, the set-up cost for the preconditioners is slightly larger than for the IC(0), while it is smaller than for the adaptive $L^T$L-projection. In the three cases, where the IC algorithm is applicable (1138bus, s1rmq4m1 and s3rmq4m1), it is the best method in one case and second best in the other, regarding both set-up cost and iteration steps. This is, although more sophisticated and efficient variants of the family of IC methods are known. The CG iteration with the matrix s3rmq4m1 preconditioned by IC(0) is interuppted because of an excessively large accumulation of round-off errors. However, since the same problem occurs for $L^T$L-projection preconditioning, the matrix s3rmq4m1 can be regarded as a difficult problem for preconditioning.
The FSAI algorithm fails in two cases (for bcsstk23 and s3rmq4m1), and the acceleration of the CG iteration is in all cases poorer than the acceleration obtained by the adaptive $L^T$L-projection algorithms. These algorithms fail only on one of the test problems, namely s3rmq4m1, caused by excessive round-off error accumulation. Although the set-up cost for the adaptive $L^T$L-projection algorithms is higher than for the FSAI algorithm, the obtained acceleration of the iterations leads to a better overall performance. Regarding the number of iterations, both pattern adaptive $L^T$L-projection algorithms perform almost the same, while the set-up cost for the $L^T$L-projection algorithm with the univariate decrease rates $\tilde{\theta}_j$ tends to be computationally cheaper than the set-up cost for the $L^T$L-projection algorithm with the decrease rates $\tilde{\lambda}_j$. Since both adaptive $L^T$L-projection algorithms are inherently parallel, these differences in the set-up cost are probably marginal in a parallel implementation.
In case of convergence, all considered iterations provide about the same reduction of the errors of the corresponding induced original linear system.

Altogether, the presented results indicate that the $L^T$L-projection algorithm with adaptive pattern derivation is in overall performance, although more expensive in the set-up cost, in efficacy comparable to the IC(0) approach. Thus, and since the $L^T$L-projection algorithm is inherently parallel, the $L^T$L-projection algorithm is a promising preconditioning technique.

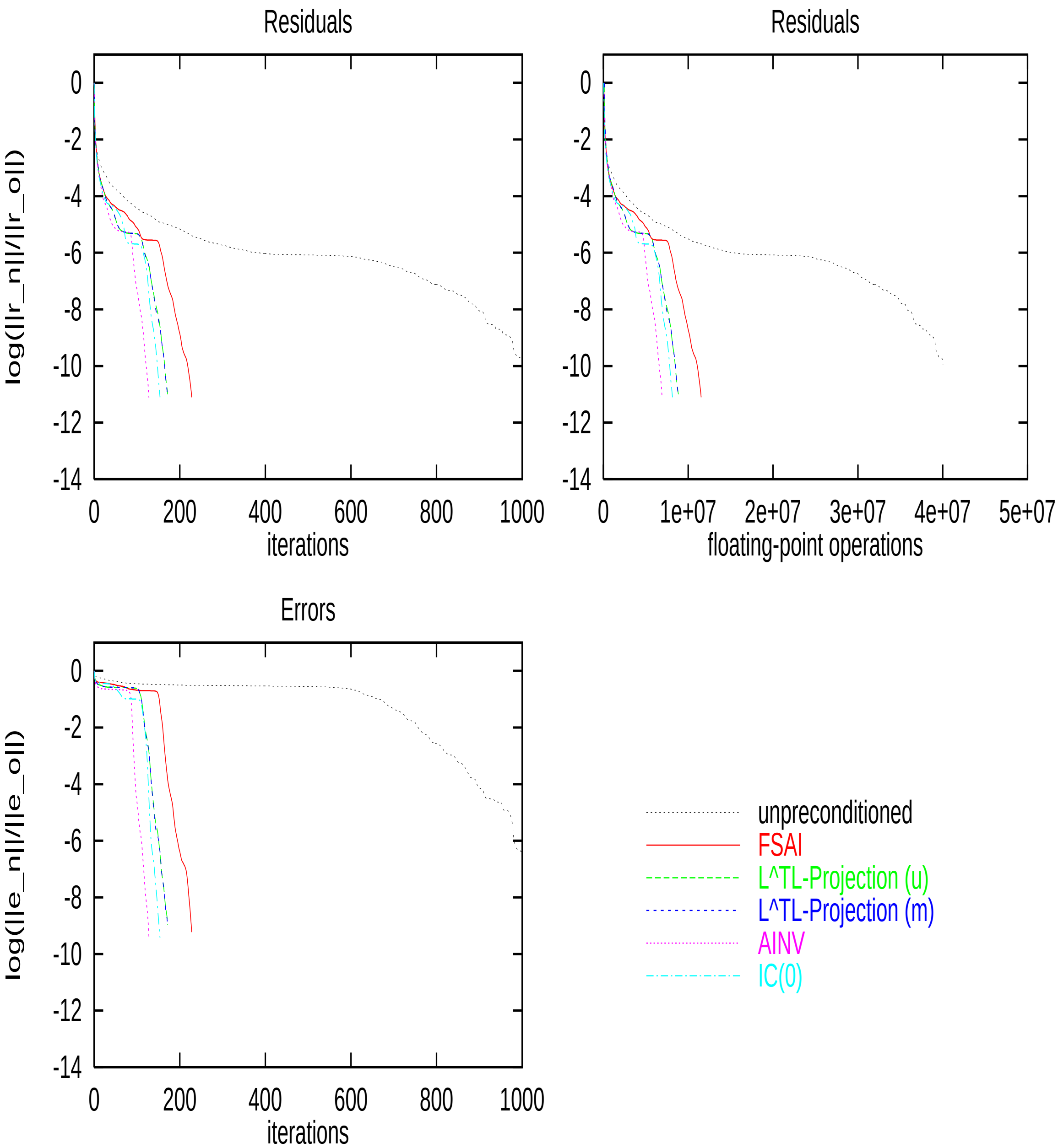


Figure 11: Comparison of the preconditioning methods for symmetric linear systems with the `CG` solver applied to the matrix `1138bus`.

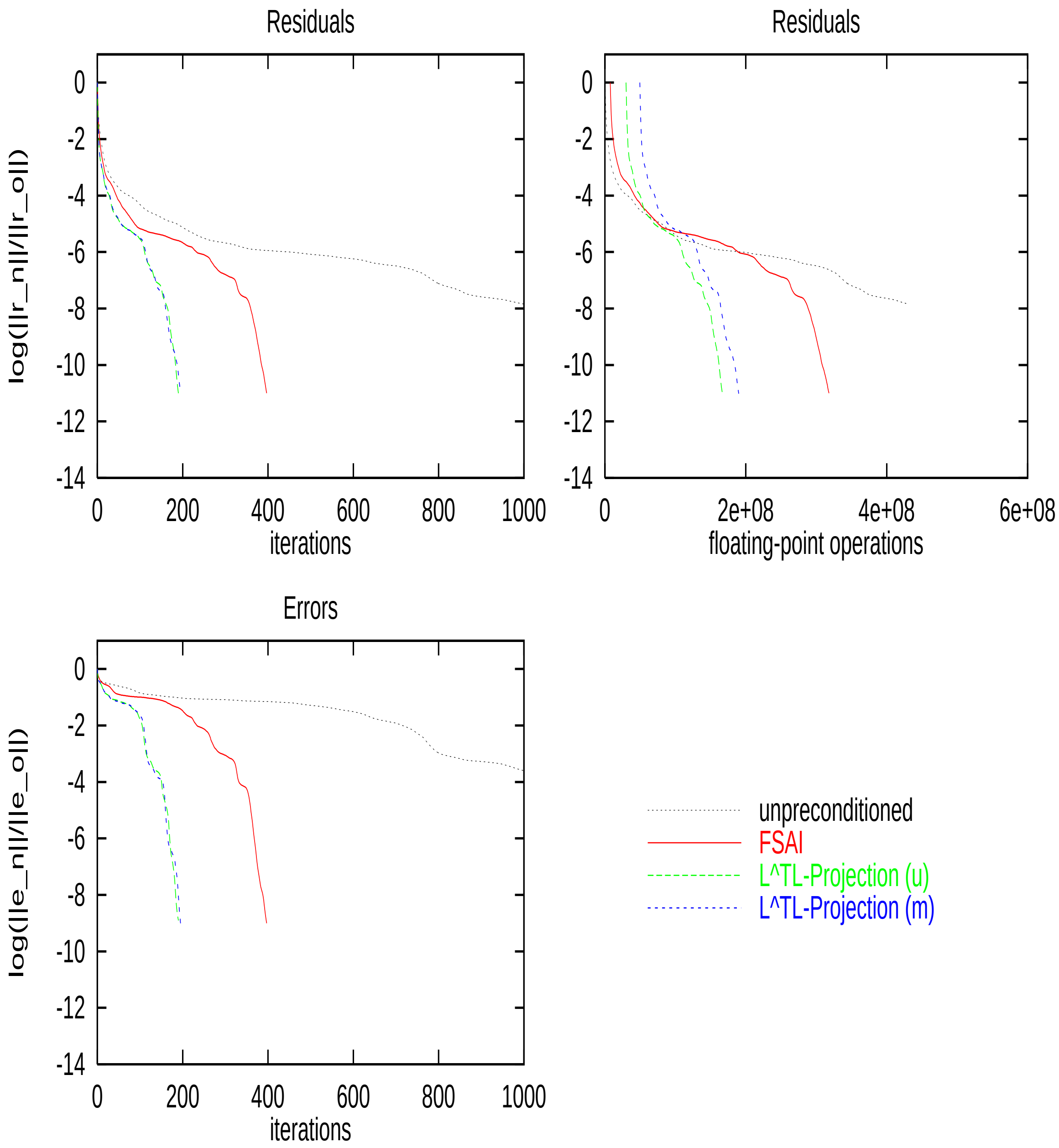


Figure 12: Comparison of the preconditioning methods for symmetric linear systems with the CG solver applied to the matrix nasa2910.

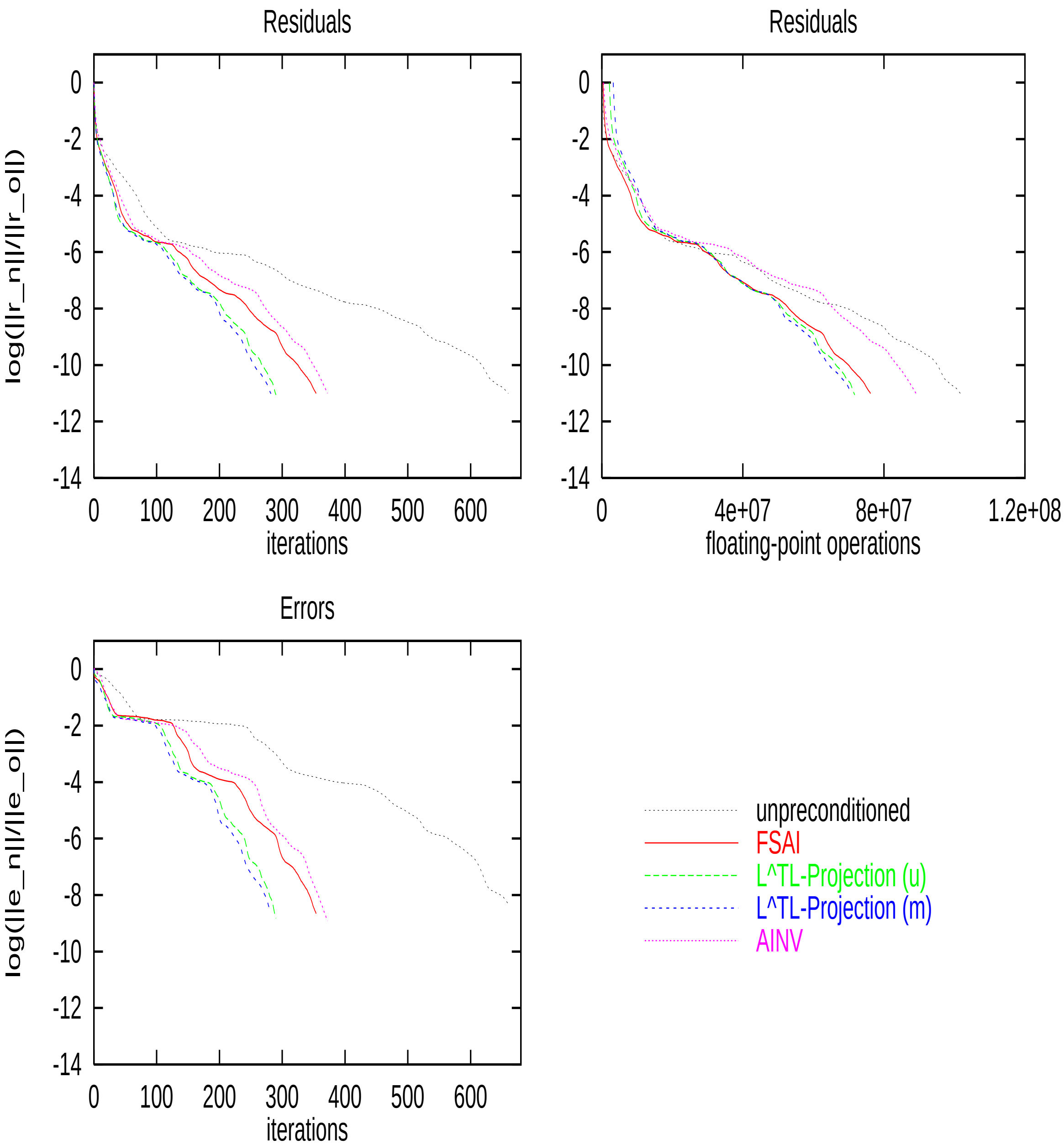


Figure 13: Comparison of the preconditioning methods for symmetric linear systems with the `CG` solver applied to the matrix `bcsstk21`.

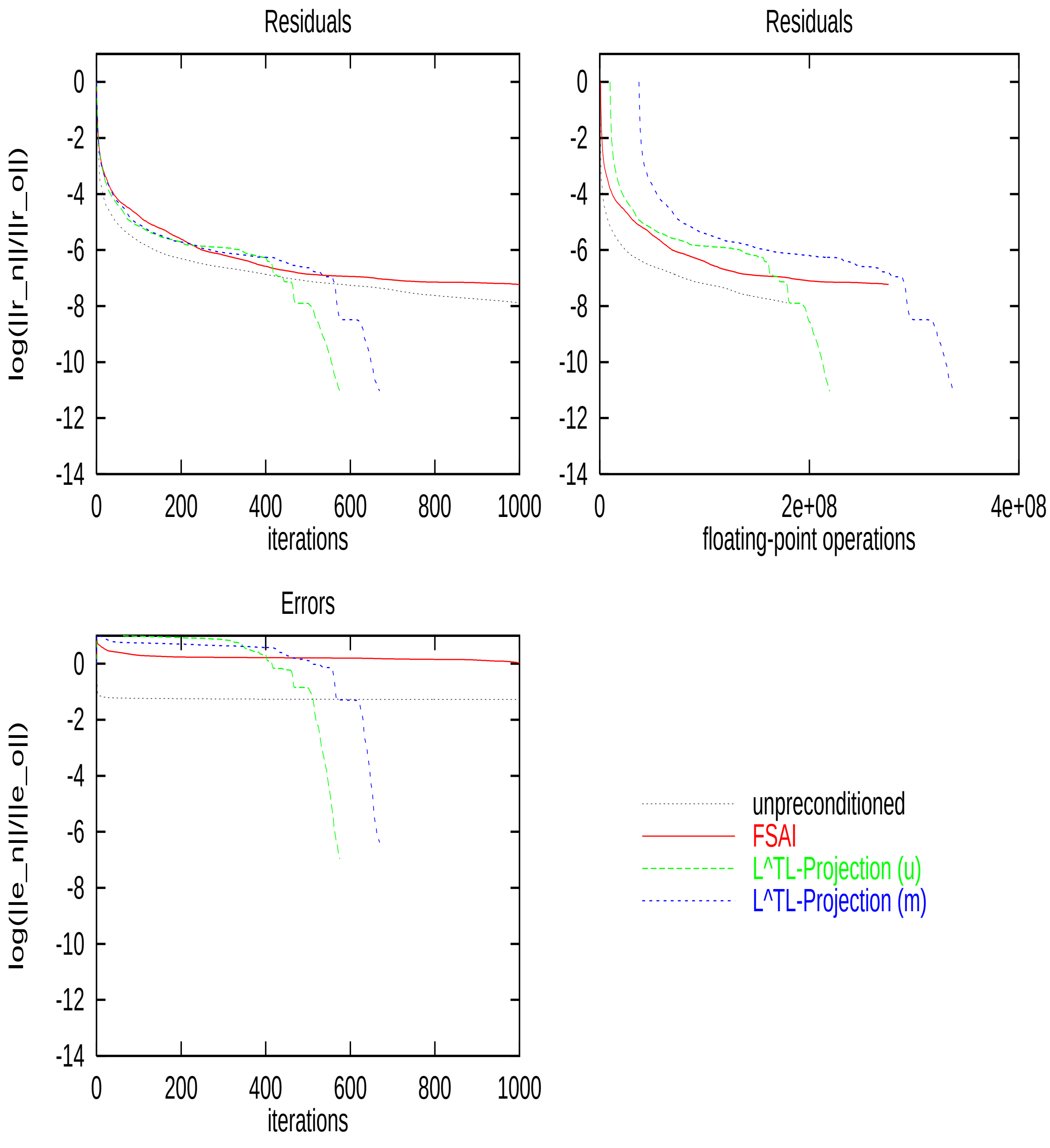


Figure 14: Comparison of the preconditioning methods for symmetric linear systems with the `CG` solver applied to the matrix `bcsstk23`.

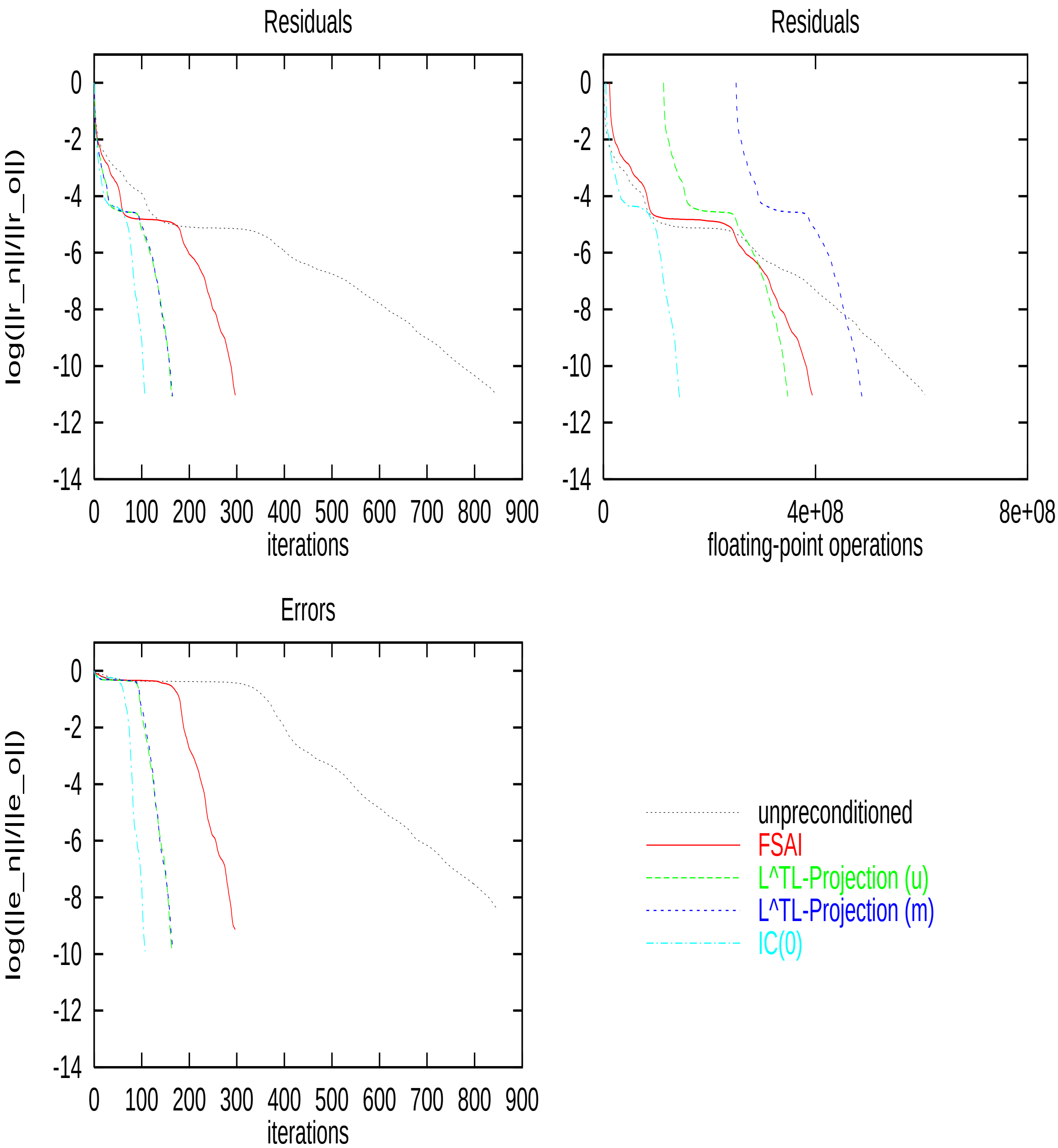


Figure 15: Comparison of the preconditioning methods for symmetric linear systems with the `CG` solver applied to the matrix `s1rmq4m1`.

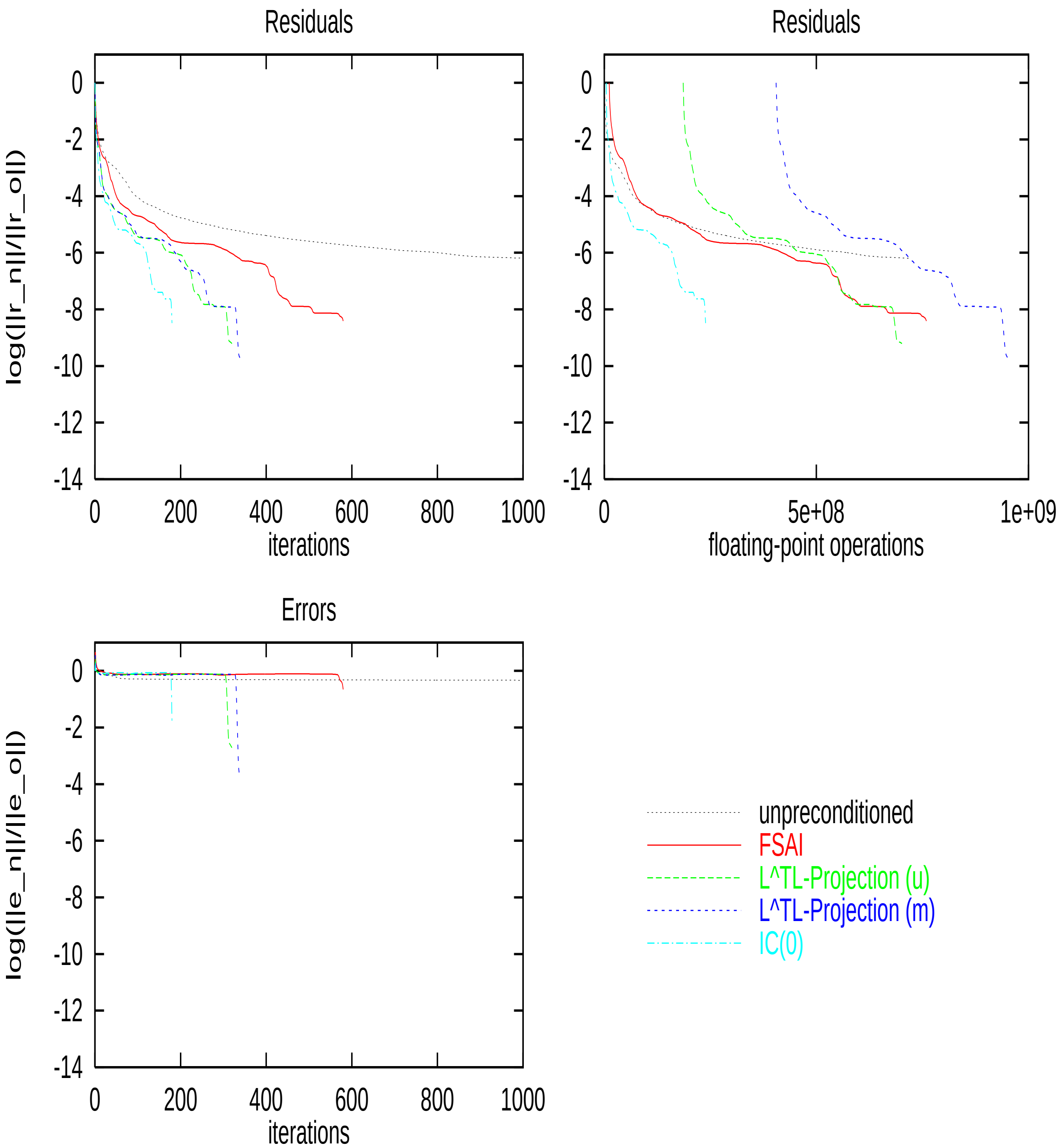


Figure 16: Comparison of the preconditioning methods for symmetric linear systems with the `CG` solver applied to the matrix `s3rmq4m1`.

## 7.5 Numerical Experiments with the Non-Symmetric Matrices

In this section, we consider the non-symmetric matrices contained in our test set. We discuss the new projection methods, i.e. `Plain projection` and `LU-projection`, in detail and we compare these new methods to some state-of-the-art preconditioning techniques.
In section 7.5.1, we discuss the `Plain projection` algorithm. In particular, we consider the efficiency of three variants of the adaptive pattern derivation and we consider the performance of the preconditioned iteration for three different iterative solvers for non-symmetric linear systems.
The `LU-projection` algorithm is considered in section 7.5.2. We consider two different strategies regarding the adaptive pattern derivation, and we consider the `FSAI` algorithm, i.e. the `LU-projection` algorithm with the prescribed projection pattern being set to the pattern of the original matrix. We compare the properties of these strategies by applying three different iterative solvers for non-symmetric linear systems.
In section 7.5.3, we compare the `Plain projection` algorithm and the `LU-projection` algorithm to some state-of-the-art preconditioning techniques for non-symmetric linear systems. The methods considered are `ILU(0)`, `AINV`, `SPAI` and `FSAI`.

### 7.5.1 The `Plain projection` Method

In this section, we discuss the numerical properties of the `Plain projection` algorithm (algorithm 13, see section 6.3) by considering the non-symmetric test problems (see section 7.2). We consider three variants for the adaptive pattern derivation, which differ in the particular utilized decrease rates.
Further, we consider three iterative solvers, namely `PRES20`, `BiCGstab` and `ATPRES` (see sections 3.5.2–3.5.4), for the preconditioned linear systems.

The first variant of the `Plain projection` algorithm is obtained by utilizing the univariate decrease rates

$$\theta_j = \frac{\left(u_j^T r_{J_k}\right)^2}{\left\|u_j\right\|_A^2}$$

from (6.26) for the pattern derivation. Note that the univariate decrease rates $\theta_j$ coincide with the estimates $\tilde{\theta}_j^2$ from (6.31) for the univariate decrease rates. For

notational convenience, we denote the resulting `Plain projection` algorithm by `us-Plain projection`.
For the second variant of the `Plain projection` algorithm, we utilize the estimated univariate decrease rates

$$\tilde{\theta}_j^1 = \frac{\left(u_j^T A^T P u_k + u_j^T r_{J_k}\right)^2}{\left\|u_j\right\|_A^2}$$

from (6.30) for the pattern derivation. We denote this variant by `ue-Plain projection`.
The third variant of the `Plain projection` algorithm is obtained by using the decrease rates

$$\lambda_j = \frac{\mid u_j^H r_{J_k} \mid^2}{\left\|u_j\right\|_A^2 - g_j^H y_j}$$

from (6.27) for the pattern derivation, where the vector $g_j \in I\!\!K^{\#J_k}$ is defined by $g_j := A(J_k, j)$, and the vector $y_j \in I\!\!K^{\#J_k}$ is the solution of the $(\#J_k \times \#J_k)$-linear system $A(J_k, J_k)y_j = g_j$. We denote this algorithm by `m-Plain projection`.

Note that the matrices employed for our numerical tests of the `Plain projection` algorithm are non-symmetric and possibly indefinite. Thus, the `Plain projection` algorithm, as deduced in section 6.3, is possibly not well defined for the non-symmetric matrices in our test set. In particular, the representation of the decrease rates $\theta_j$ and $\lambda_j$, as stated by Corollary 6.44 requires the matrix $A$ to be symmetric. Further, the candidate sets from definition 5.1 are not known a priori. Thus, in our implementation we simply determine the particular decrease rates for all indices $j$ which are not elements of the current column-pattern $J_k$. In spite of this scruples, in all our numerical tests, a singular inner system never occurred.
Note that, if the matrix $A \in I\!\!K^{n \times n}$ is indefinite, the expression $\left\|x\right\|_A^2$ is not a vector-norm for $x \in I\!\!K^n$, but the mnemonic abbreviation of $x^H A x$. As a consequence, the decrease rates $\theta_j$, $\tilde{\theta}_j^1$ and $\lambda_j$ may be negative. Thus, for choosing the new indices in the pattern derivation process it is possible both to refer to the actual values or to the absolute values of the decrease rates $\theta_j$, $\tilde{\theta}_j^1$ and $\lambda_j$.
Our numerical tests done in the course of this work indicate, that for the pattern derivation process always the absolute values of the decrease rates $\theta_j$, $\tilde{\theta}_j^1$ or $\lambda_j$ should be applied.
The initial projection pattern for all three variants of the `Plain projection` algorithm is always the diagonal pattern, i.e. $J_k^0 := (k)$ for $k = 1, \ldots, n$. Since all of the matrices in our test set have a zero-free diagonal, for this choice of the initial projection pattern the first step of the pattern derivation of the considered `Plain projection` algorithms is always practical according to definition 4.3. Note that utilizing diagonal intial projection patterns is no restriction because matrices with

zeros on the main diagonal can easily be permuted to have a zero-free main diagonal, see e.g. [23].
For the `Plain projection` algorithm, four control parameters for the pattern derivation must be supplied (those are explained in detail for algorithm 8, see pages 68–69). Those parameters are

i) $\underline{mf}$ : The maximum number of non-zeros in each column of the projective approximate inverse.

ii) $\underline{\epsilon_k}$ : The threshold value for the qualitative stopping criterion.

iii) $\underline{ms}$ : The maximum number of pattern updating steps for each column-pattern.

iv) $\underline{mfps}$ : The maximum number of indices added in one updating step to the column-pattern.

In all tests of the `Plain projection` algorithms, we utilize the combined stopping criterion. We tried to select the above parameters in such a way that the obtained projective approximate inverses have at most twice as many non-zeros as the corresponding original matrices. The particular choices for the parameters $mf$, $\epsilon_k$, $ms$, $mfps$, as stated in table 13, are obtained after a few test runs of each test problem. The sensitivity of the `Plain projection` algorithms to changes of these parameters depends on the tests problems and is not known a priori.

**Comparison of the `Plain projection` Algorithms**
The performance results of the `Plain projection` algorithms are stated in table 14. The column $\mathrm{nz}(P)$ this table contains the number of non-zeros in the particular projective approximate inverses. For the `BiCGstab` iteration, we state, in addition to the number of iterations, the final error of the induced iteration for the original linear systems (see definition 3.1).
First of all, we note that the performance of the three iterative methods shows drastical differences. The `BiCGstab` method is by far the most robust iterative solver. It fails only on eight out of 39 test runs. The `PRES20` method fails on 21 test runs. The reason for this is probably the truncation which forces a short recurrence for this method (see page 33, section 3.5.2 and the references therein). The `ATPRES` iteration fails on 31 out of 39 test runs. This is probably due to the fact that this iterative method is relatively stable, but slow in convergence (see page 33, section 3.5.4 and the references therein). The large number of diverging test runs suggests that the `ATPRES` iteration does not collaborate with the `Plain projection` preconditioning technique.
We consider the results for the `BiCGstab` iteration in greater detail: the `us-Plain projection` and the `ue-Plain projection` fail only in two cases (for the matrices

l_50_1000 and c100000m1o4), while the `m-Plain projection` additionally fails for the matrices sherman2 and c500m1o4. In case of convergence, the acceleration of the `BiCGstab` iteration, the Euclidian norm of the final error, and the number of non-zeros in the projective approximate inverses, is, in tendency, about the same for all three `Plain projection` algorithms. As stated by the plots in figures 17–28, the computational complexity (i.e. the number of floating-point operations) of the three `Plain projection` algorithms shows considerable differences. In tendency, the cheapest variant is the `us-Plain projection` algorithm, while the computational cost for the `ue-Plain projection` algorithm is only slightly higher. In general, the computational complexity for the `m-Plain projection` algorithm is by far the highest of the three variants. The computational complexity for all three variants of the `Plain projection` algorithm is dominated by the solving the inner linear systems.
All three variants of the `Plain projection` algorithm fail on the matrices l_50_1000 and c100000m1o4. Interestingly, this does not necessarily indicate that these matrices cannot be preconditioned by projective approximate inverses. Conversely, the projective approximate inverse of the matrix l_50_1000 determined on the fixed projection pattern equal to the original pattern contains only 12300 non-zeros and leads to rapid convergence (see figure 26). However, the same strategy fails to work for the matrix c100000m1o4.
The convergence behavior of the preconditioned iterations, as summarized in table 14, is visualized in figures 17–28. In these figures, the curves denoted by the "us_" prefix correspond to the `us-Plain projection`. The curves with the "ue_" prefix are obtained by the `ue-Plain projection`, and the curves prefixed by "m_" correspond to the `m-Plain projection`.

The results for the three `Plain projection` algorithms, as stated by table 14 and by figures 17–28, indicate that the `Plain projection` preconditioning techniques substantially accelerate the `BiCGstab` iteration. For a comparison to the corresponding results for the unpreconditioned iterations see tables 3, 5 and 9.

| matrix | decrease rates | Plain projection | | | | nz($P$) | PRES20 | BiCGstab | | ATPRES |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $mf$ | $ms$ | $mfps$ | $\epsilon_k$ | | its | its | $\|\|x^* - x_k\|\|_2$ | its |
| orsirr2 | us,ue,m | 6 | 5 | 1 | 0.5 | 3009 | 96 | 51 | $9.6 \cdot 10^{-9}$ | ⊘ |
| pores2 | us | 13 | 6 | 2 | 0.4 | 7781 | 650 | 128 | $4.4 \cdot 10^{-10}$ | ⊘ |
| | ue | 13 | 6 | 2 | 0.5 | 7303 | ⊘ | 126 | $1.2 \cdot 10^{-8}$ | ⊘ |
| | m | 13 | 6 | 2 | 0.5 | 6495 | ⊘ | 118 | $2.2 \cdot 10^{-8}$ | ⊘ |
| sherman2 | us | 40 | 13 | 3 | 0.3 | 17778 | ⊘ | 35 | $1.6 \cdot 10^{-9}$ | ⊘ |
| | ue | 31 | 10 | 3 | 0.33 | 15210 | 347 | 70 | $4.9 \cdot 10^{-10}$ | ⊘ |
| | m | | | | | | ⊘ | ⊘ | | ⊘ |
| saylr4 | us | 21 | 10 | 2 | 0.3 | 42544 | ⊘ | 354 | $1.8 \cdot 10^{-6}$ | ⊘ |
| | ue | 21 | 10 | 2 | 0.3 | 43156 | ⊘ | 329 | $1.8 \cdot 10^{-6}$ | ⊘ |
| | m | 15 | 7 | 2 | 0.3 | 38568 | ⊘ | 889 | $2.6 \cdot 10^{-6}$ | ⊘ |
| memplus | us | 11 | 5 | 2 | 0.75 | 33808 | 392 | 372 | $7.1 \cdot 10^{-7}$ | ⊘ |
| | ue | 11 | 5 | 2 | 0.75 | 37130 | 344 | 562 | $3.9 \cdot 10^{-7}$ | ⊘ |
| | m | 11 | 5 | 2 | 0.75 | 33826 | 363 | 162 | $3.7 \cdot 10^{-7}$ | ⊘ |
| utm1700b | us | 15 | 7 | 2 | 0.3 | 20050 | ⊘ | 358 | $2.2 \cdot 10^{-7}$ | ⊘ |
| | ue | 15 | 7 | 2 | 0.3 | 21078 | ⊘ | 385 | $9.5 \cdot 10^{-8}$ | ⊘ |
| | m | 15 | 7 | 2 | 0.3 | 20062 | ⊘ | 346 | $1.2 \cdot 10^{-7}$ | ⊘ |
| utm3060 | us | 19 | 9 | 2 | 0.4 | 42069 | ⊘ | 401 | $5.0 \cdot 10^{-7}$ | ⊘ |
| | ue | 19 | 9 | 2 | 0.4 | 45796 | ⊘ | 847 | $5.7 \cdot 10^{-7}$ | ⊘ |
| | m | 19 | 9 | 2 | 0.4 | 42254 | ⊘ | 336 | $3.4 \cdot 10^{-7}$ | ⊘ |
| l_50_1 | us,m | 6 | 5 | 1 | 0.3 | 14407 | 224 | 73 | $7.4 \cdot 10^{-9}$ | 566 |
| | ue | 5 | 4 | 1 | 0.3 | 12295 | 211 | 81 | $3.4 \cdot 10^{-10}$ | 883 |
| l_50_100 | us | 11 | 5 | 2 | 0.5 | 25754 | ⊘ | 53 | $8.7 \cdot 10^{-11}$ | ⊘ |
| | ue | 11 | 5 | 2 | 0.5 | 26026 | 76 | 35 | $5.7 \cdot 10^{-11}$ | 217 |
| | m | 11 | 5 | 2 | 0.5 | 25754 | 145 | 57 | $3.9 \cdot 10^{-13}$ | 539 |
| l_50_1000 | us,ue,m | | | | | | ⊘ | ⊘ | | ⊘ |
| | fixed | | | | | 12300 | 283 | 156 | $1.2 \cdot 10^{-9}$ | ⊘ |
| c1m1o4 | us | 5 | 2 | 2 | 0.3 | 92856 | 144 | 61 | $3.3 \cdot 10^{-8}$ | 362 |
| | ue | 5 | 2 | 2 | 0.3 | 92928 | 147 | 54 | $1.0 \cdot 10^{-8}$ | 341 |
| | m | 5 | 2 | 2 | 0.3 | 92856 | 144 | 55 | $1.5 \cdot 10^{-8}$ | 362 |
| c500m1o4 | us | 11 | 10 | 1 | 0.3 | 189357 | 635 | 264 | $9.4 \cdot 10^{-9}$ | ⊘ |
| | ue | 11 | 10 | 1 | 0.3 | 195374 | 584 | 173 | $3.4 \cdot 10^{-9}$ | ⊘ |
| | m | | | | | | ⊘ | ⊘ | | ⊘ |
| c100000m1o4 | us,ue,m | | | | | | ⊘ | ⊘ | | ⊘ |

Table 14: Performance of the solvers applied to the unsymmetric linear systems precondititioned by the `Plain projection` algorithm

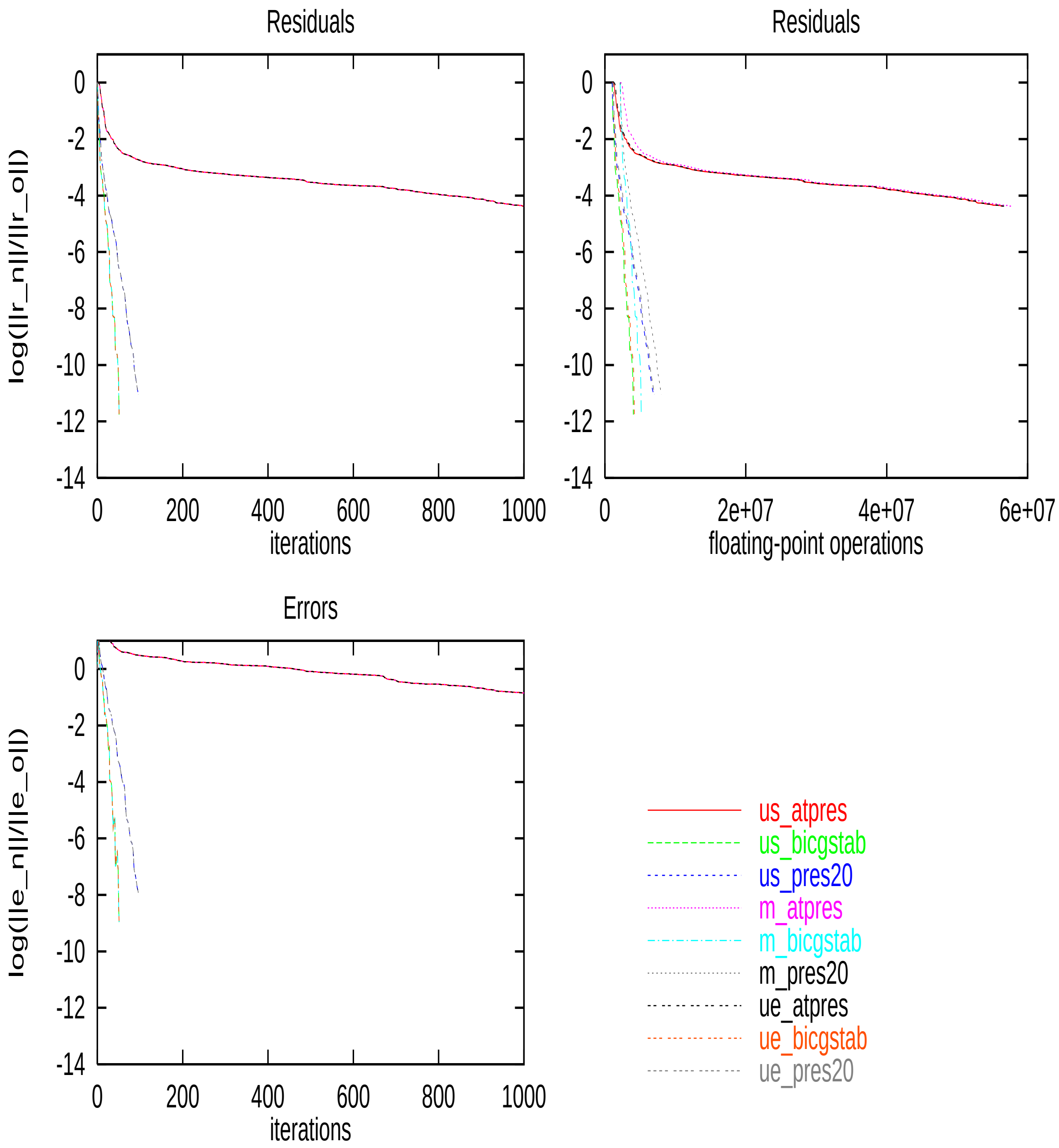


Figure 17: Convergence of the unsymmetric solvers precondititioned by the `Plain projection` method for the matrix orsirr2.

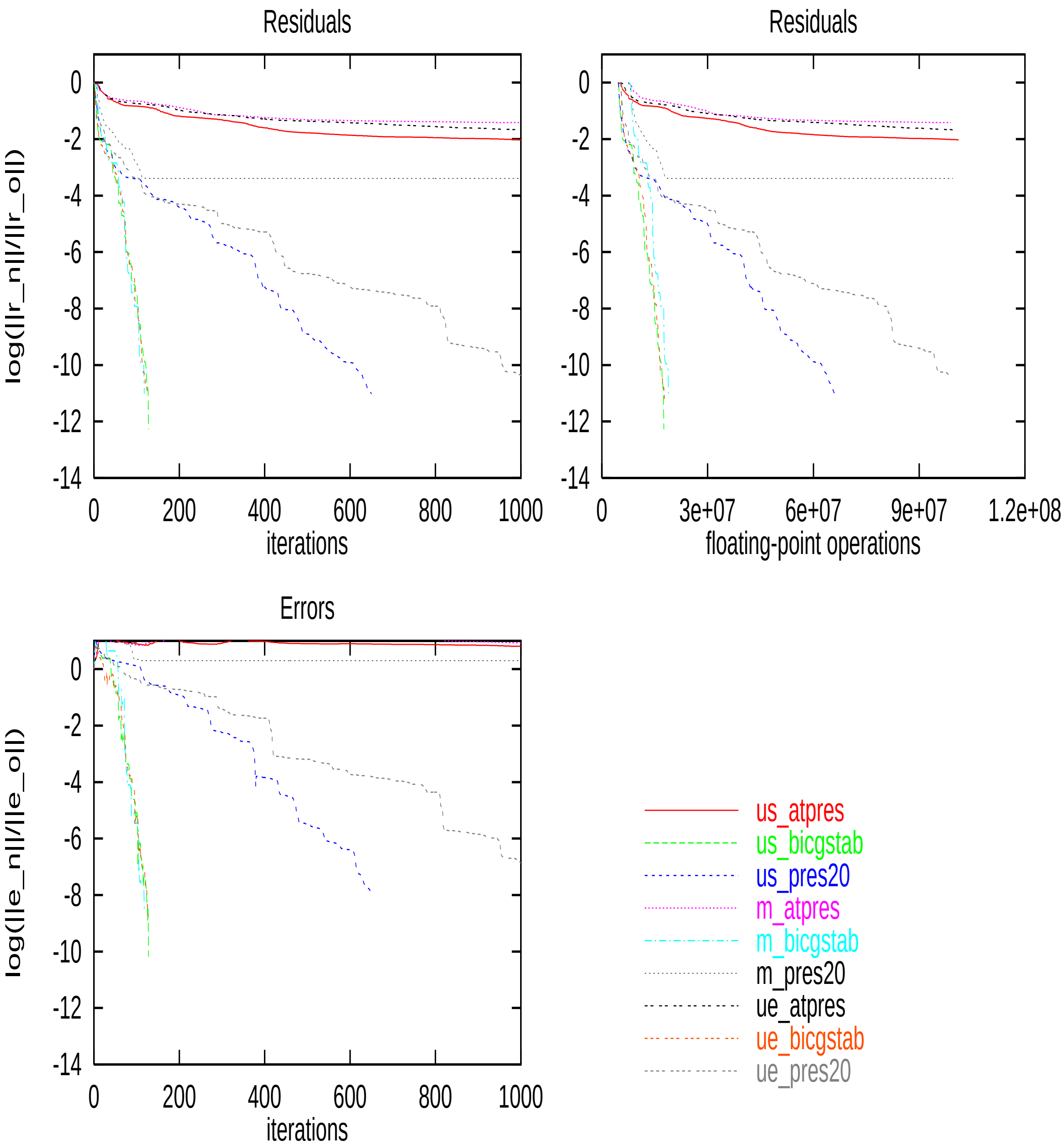


Figure 18: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix pores2.

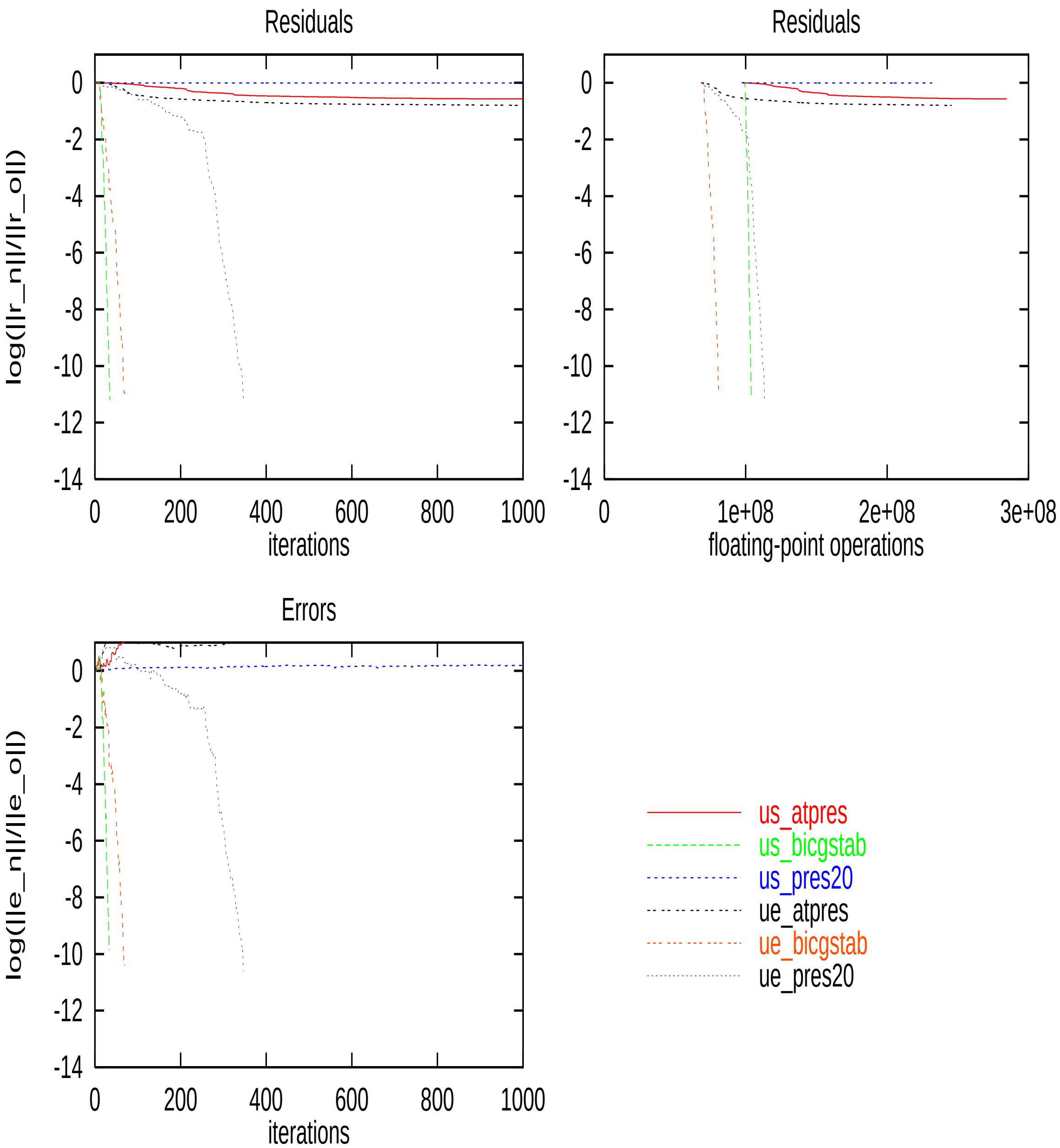


Figure 19: Convergence of the unsymmetric solvers precondititioned by the `Plain projection` method for the matrix sherman2.

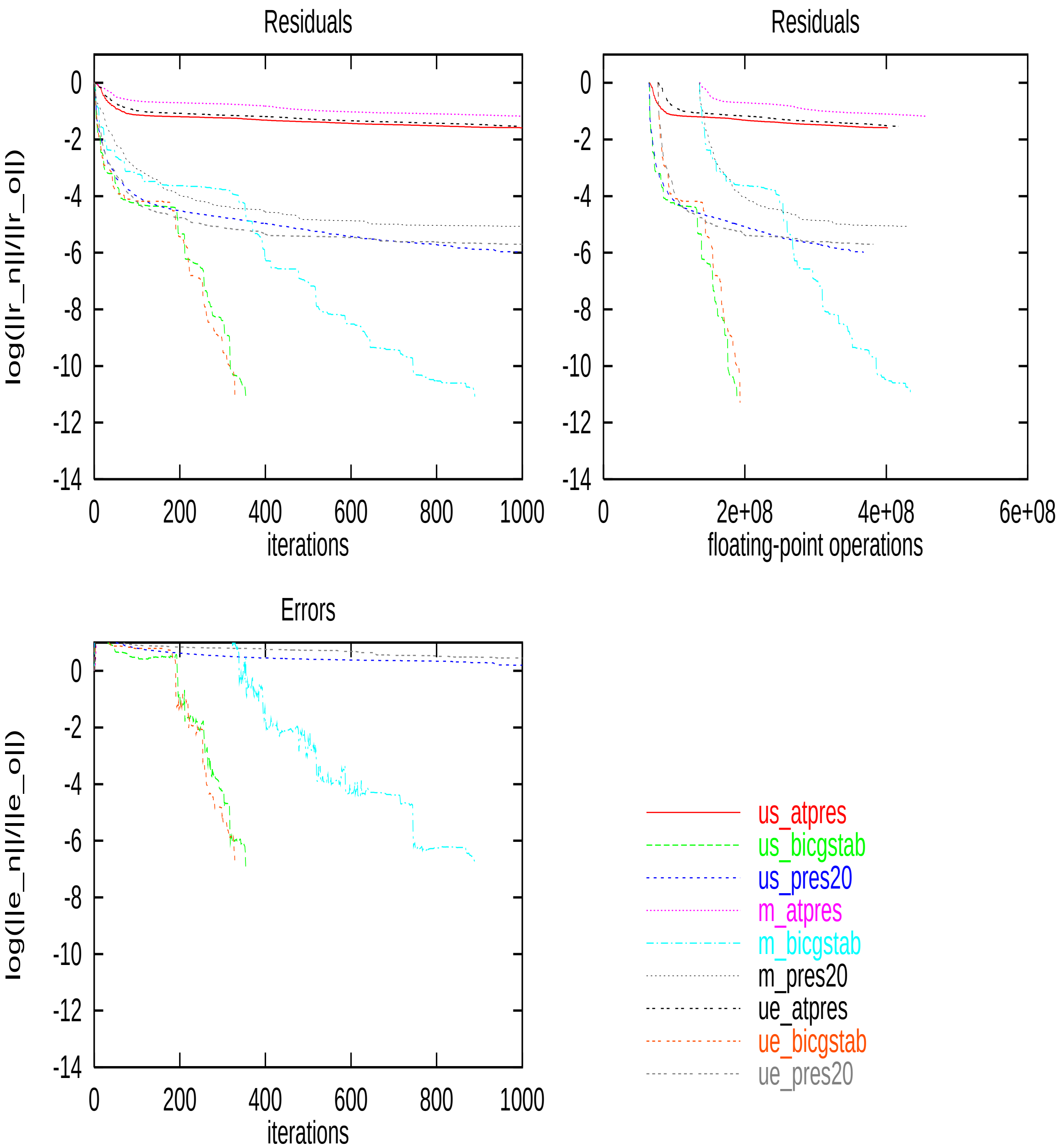


Figure 20: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix saylr4.

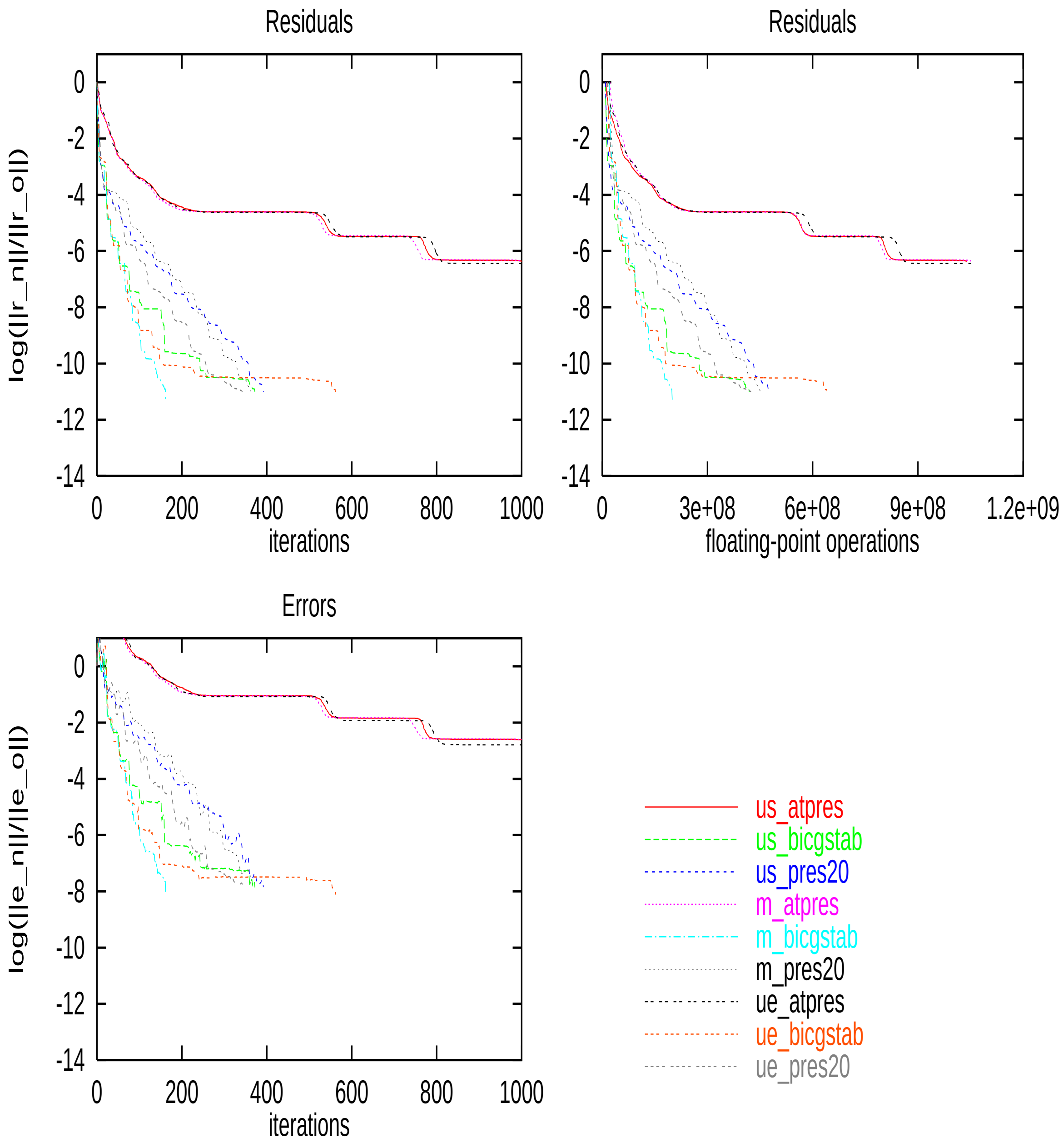


Figure 21: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix memplus.

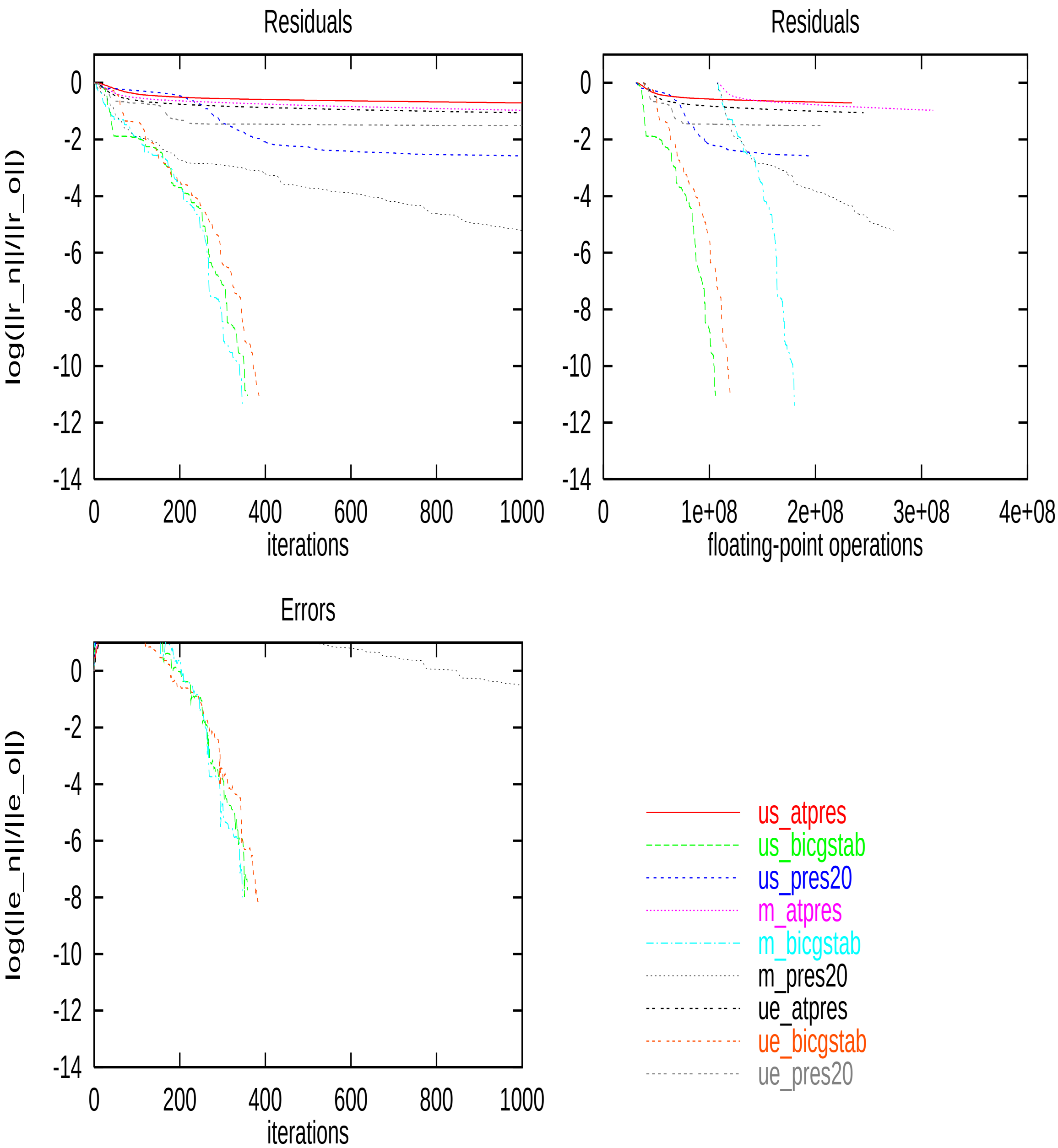


Figure 22: Convergence of the unsymmetric solvers precondititioned by the `Plain projection` method for the matrix `utm1700b`.

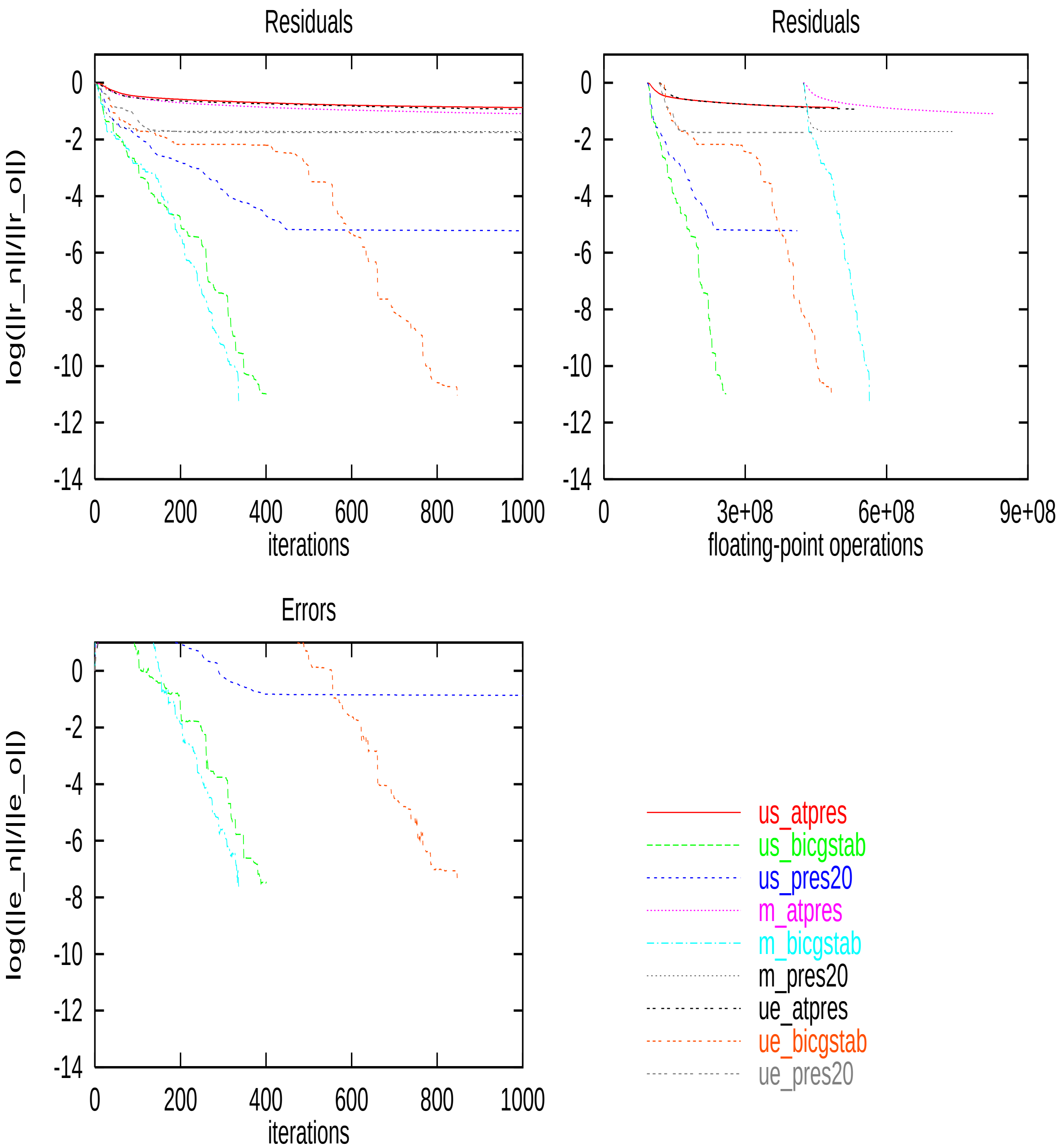


Figure 23: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix `utm3060`.

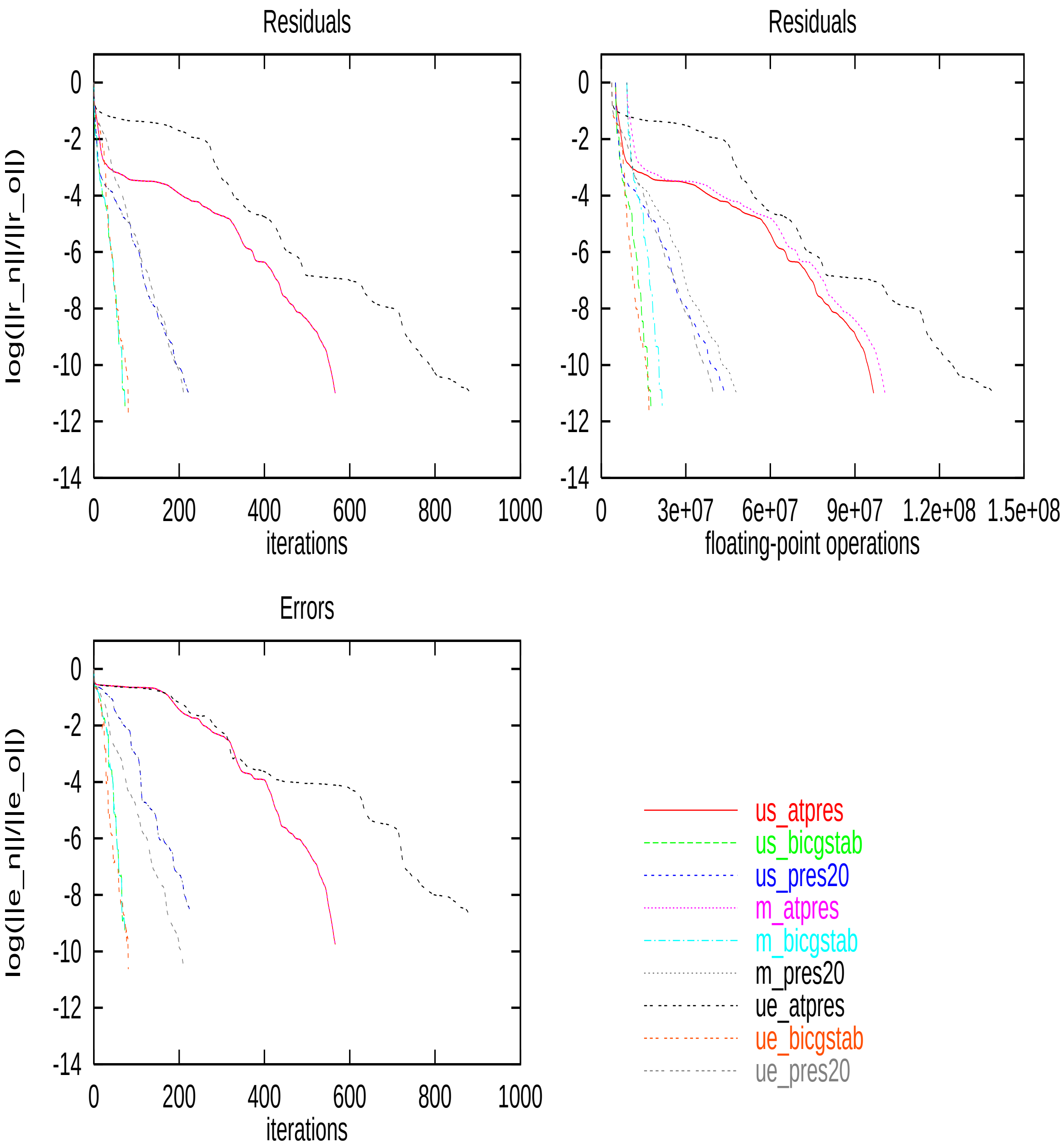


Figure 24: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix l_50_1.

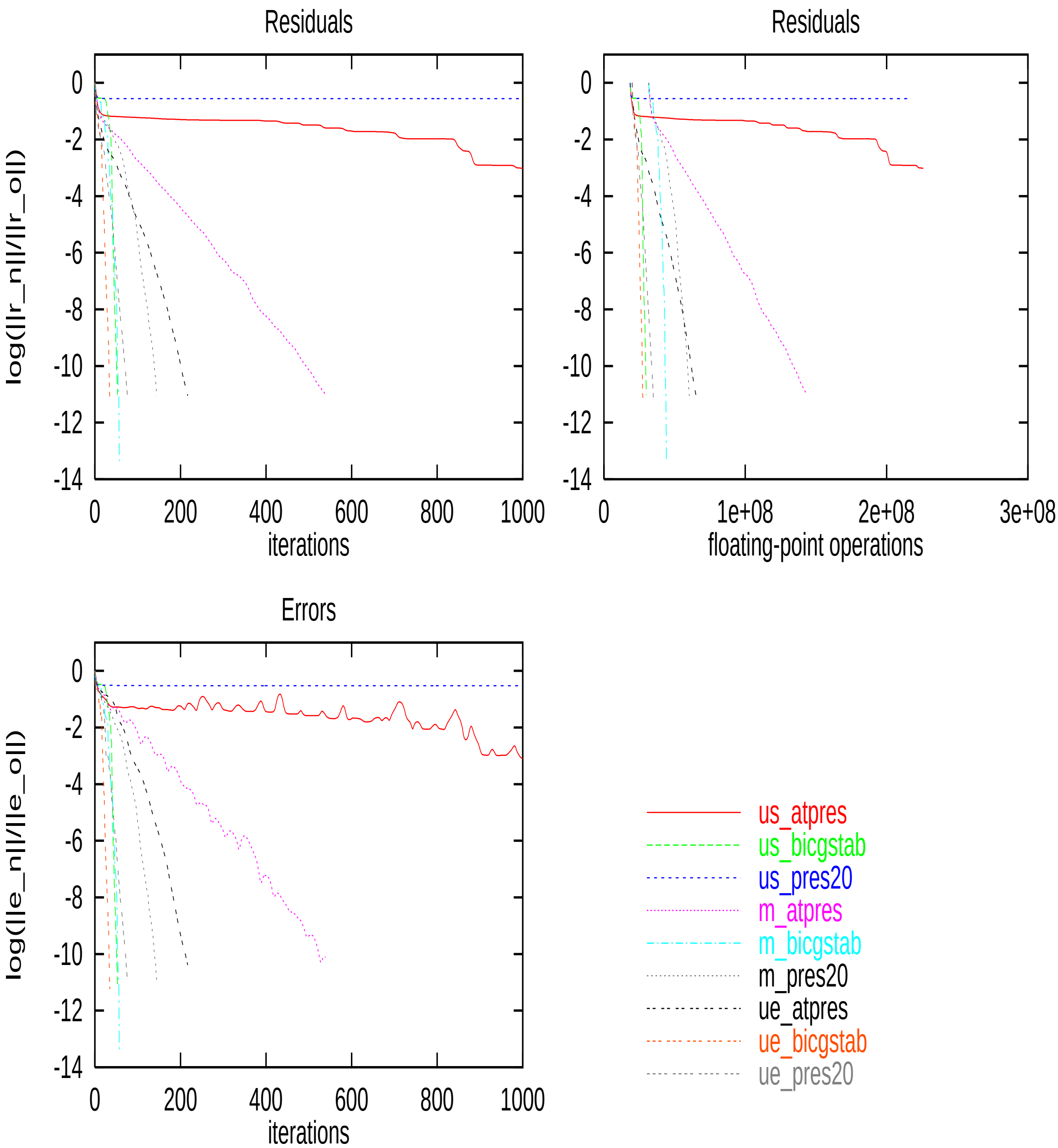


Figure 25: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix l_50_100.

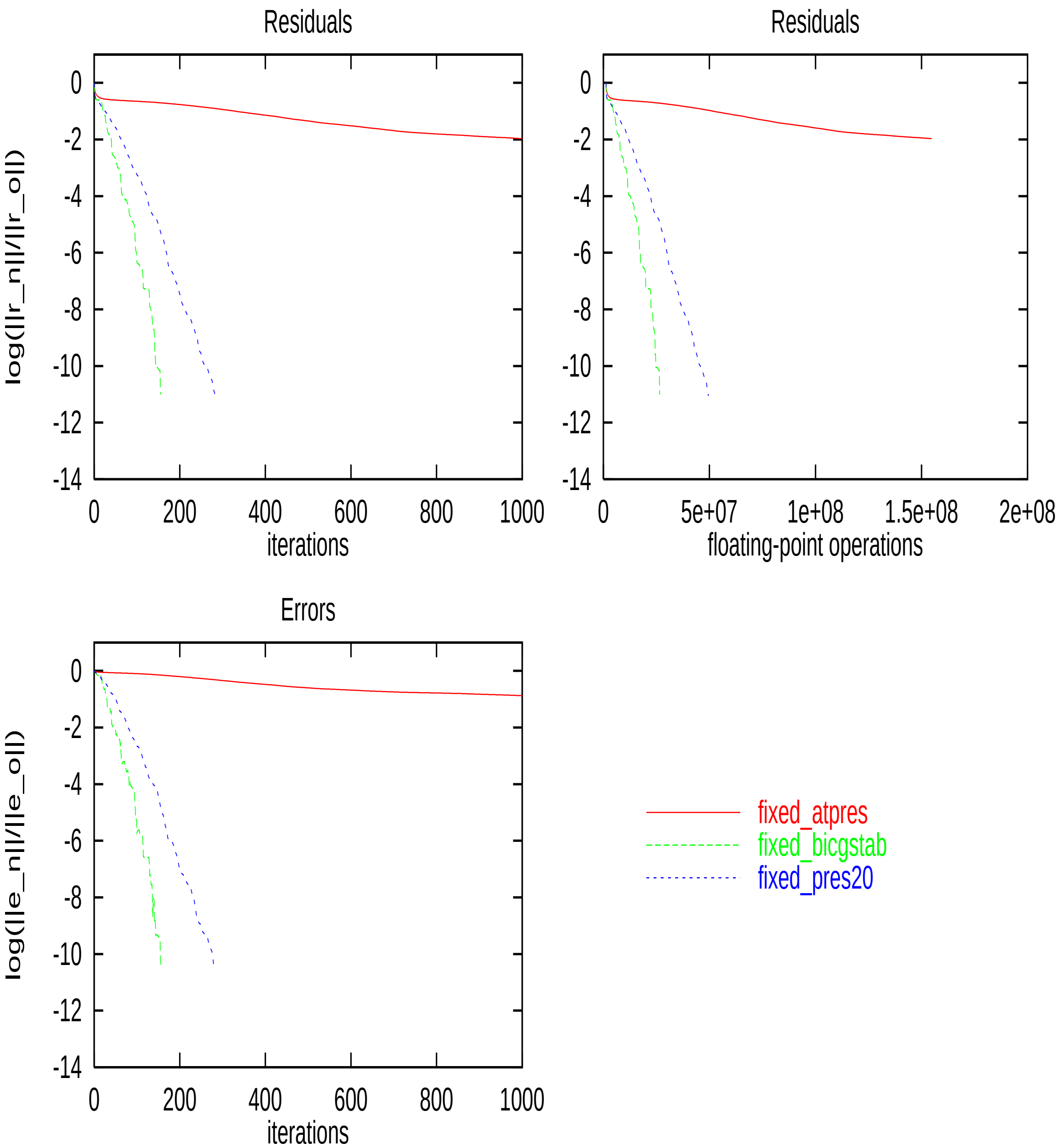


Figure 26: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method applied to the matrix l_50_1000.

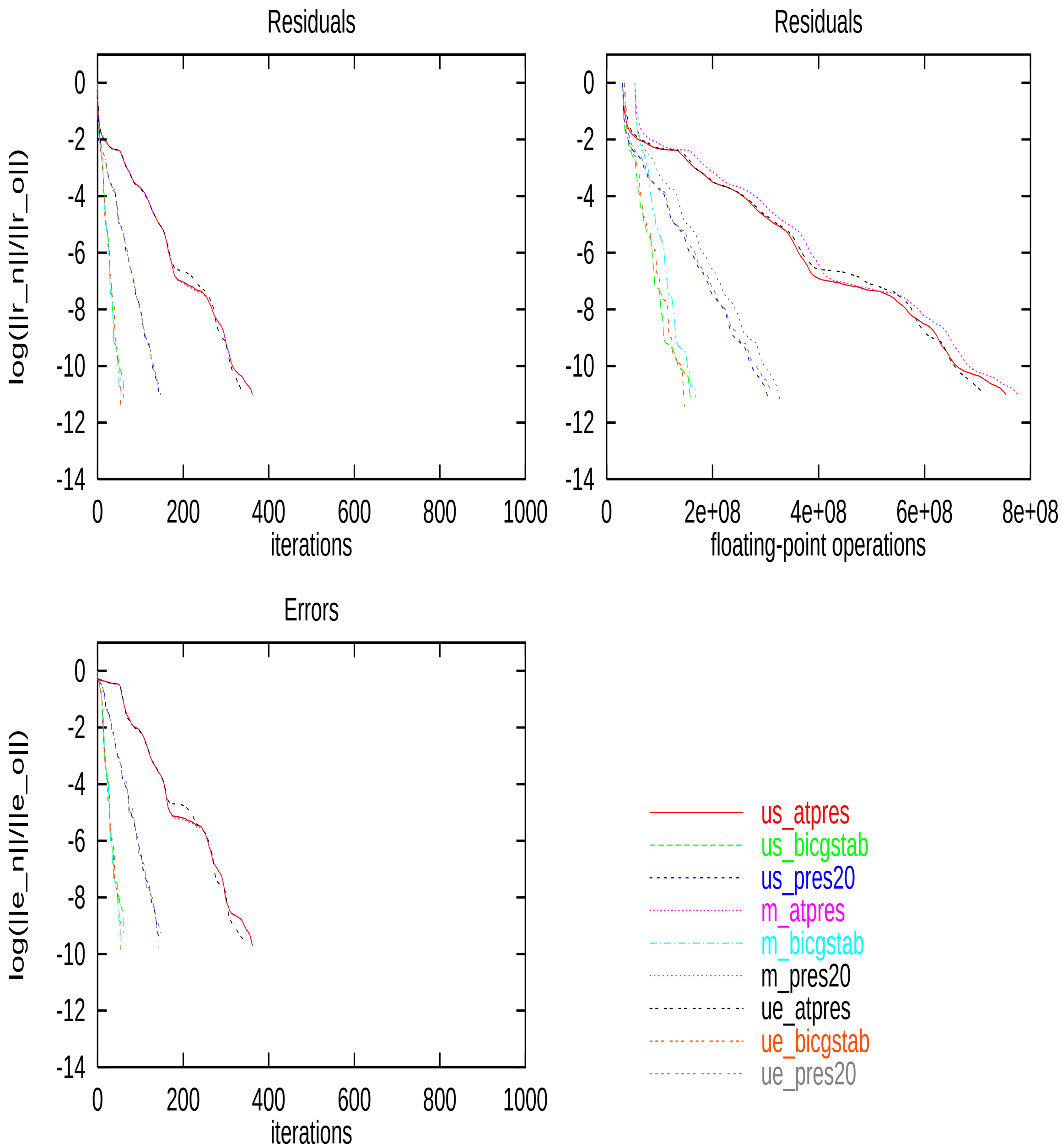


Figure 27: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix `c1m1o4`.

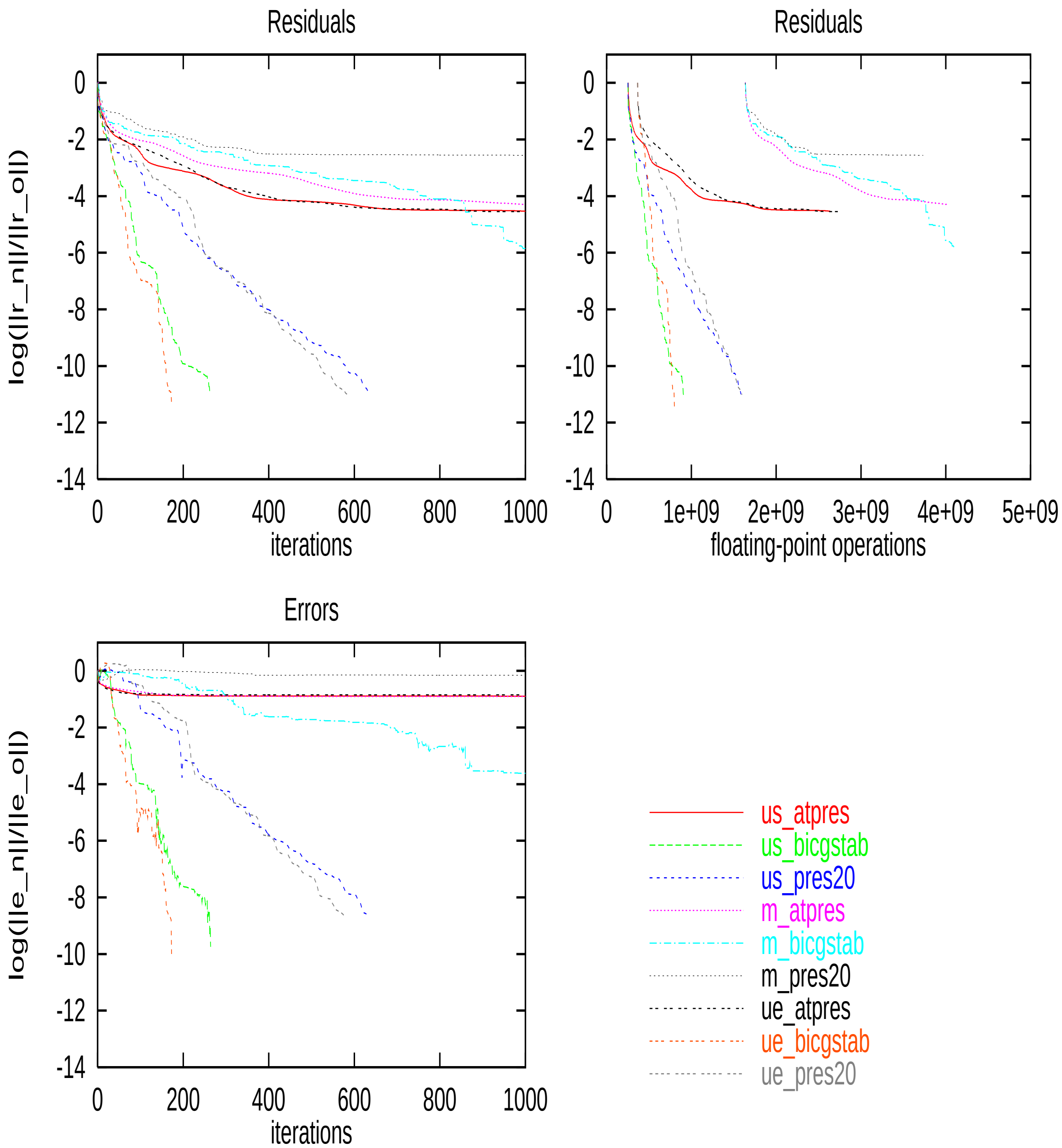


Figure 28: Convergence of the unsymmetric solvers preconditioned by the `Plain projection` method for the matrix `c500m1o4`.

### 7.5.2 The `LU-projection` algorithm

In this section, we discuss the numerical properties of the `LU-projection` algorithm derived in section 6.7.2.
We consider three variants of this algorithm. The first variant prescribes the projection pattern to equal the pattern of the original matrix. This method is denoted by `FSAI`. Further, we consider two new pattern adaptive variants of the `LU-projection` algorithm. These algorithms differ in the utilized estimates for the univariate decrease rates. The first pattern adaptive variant, denoted by `us-LU-projection`, utilizes the estimates $\tilde{\theta}_j^2$ and $\tilde{\theta}_j^4$, defined in (6.70) and (6.72). The second variant, denoted by `ue-LU-projection`, uses the estimates $\tilde{\theta}_j^1$ and $\tilde{\theta}_j^3$, defined in (6.69) and (6.71), for the pattern derivation.
For reasons of simplicity, we use the same parameters for controlling the pattern derivation for both triangular projective approximate inverses. The performance of these three considered variants of the `LU-projection` algorithm is summarized in table 15. The parameter choices, as shown in table 15, are obtained after a few test runs for the pattern adaptive `LU-projection` variants. In table 15, the rows with the entry "fixed" in the column "decrease rates" contain the results obtained by `FSAI` preconditioning; the rows with the entries "`us`" and "`ue`" contain the results of the `us-LU-projection` algorithm and of the `ue-LU-projection` algorithm, respectively. The columns nz($L$) and nz($U$) in this table contain the number of non-zeros in the lower and in the upper triangular projective approximate inverse.

**Comparison of the `LU-projection` Algorithms**
Similar to the results for the `Plain projection` algorithm, the differences in the performance of the iterative solvers are drastical. The `BiCGstab` iteration is by far the most stable; it fails only in 13 out of 39 tests runs. The `PRES20` iteration fails in 24 cases and the `ATPRES` iteration fails in 33 cases.
We consider the performance of the `BiCGstab` iteration in greater detail. The `FSAI` algorithm fails on seven out of the 13 test problems, while the pattern adaptive algorithms `us-LU-projection` and `ue-LU-projection` fail on three test problems only. In six out of the seven converging tests with the `FSAI` algorithm, the corresponding unpreconditioned linear systems converged as well. Thus, the `FSAI` algorithm is not a robust preconditioning technique.
The performance of the two pattern adaptive variants `us-LU-projection` and `ue-LU-projection` is comparable, while the set-up cost for `ue-LU-projection` is, in tendency, higher than for `us-LU-projection`. The number of non-zeros in the triangular projective approximate inverses with adaptively determined projection patterns is limited to four times the number of non-zeros in the original matrix. In all cases, the number of non-zeros in the triangular factors for the patterns adaptive `LU-projection` variants is higher than the number of non-zeros for the `Plain`

`projection` variants.
In figures 29–38, the plots of the preconditioned iterations are shown. The curves prefixed by "`FSAI_`" are obtained by `FSAI` preconditioning, and the curves prefixed by "us_" and "ue_" correspond to `us-LU-projection` and to `ue-LU-projection`, respectively.

Altogether, the results for `FSAI` indicate, that this variant is not recommendable. The pattern adaptive variants `us-LU-projection` and `ue-LU-projection`, are capable of substantially accelerating the `BiCGstab` iteration.
The required number of non-zeros in the triangular projective approximate inverses is considerable larger than for the `Plain projection` algorithms, while the acceleration of the iterative solve is somewhat less than for `Plain projection`.
The efficiency of `LU-projection` may be promoted by permuting the linear system prior to applying the `LU-projection` algorithms. In [7], it is surveyed that various permutations may promote the efficiency of incomplete factorizations (e.g. `ILU`). Analogous results are reported in [10] for the `AINV` preconditioning technique.

| matrix | decrease rates | `LU-projection` $mf$ | $ms$ | $mfps$ | nz($P_L$) | nz($P_U$) | `PRES20` its | `BiCGstab` its | $\|\|x^* - x_k\|\|_2$ | `ATPRES` its |
|---|---|---|---|---|---|---|---|---|---|---|
| orsirr2 | fixed | – | – | – | 3428 | 3428 | 407 | 189 | $1.8 \cdot 10^{-8}$ | ⊘ |
| | `us` | 6 | 5 | 1 | 5224 | 5227 | 85 | 37 | $1.2 \cdot 10^{-8}$ | ⊘ |
| | `ue` | 6 | 5 | 1 | 5224 | 5227 | 80 | 36 | $3.1 \cdot 10^{-9}$ | ⊘ |
| pores2 | fixed | – | – | – | 5368 | 4826 | ⊘ | 377 | $1.1 \cdot 10^{-8}$ | ⊘ |
| | `us` | 11 | 5 | 2 | 12643 | 12219 | 371 | 62 | $1.1 \cdot 10^{-8}$ | ⊘ |
| | `ue` | 11 | 5 | 2 | 11999 | 9388 | ⊘ | 101 | $6.8 \cdot 10^{-9}$ | ⊘ |
| sherman2 | fixed | – | – | – | 10952 | 13098 | ⊘ | ⊘ | | ⊘ |
| | `us` | 40 | 13 | 3 | 30779 | 40718 | ⊘ | 119 | $5.8 \cdot 10^{-8}$ | ⊘ |
| | `ue` | 31 | 10 | 3 | 23794 | 30108 | 182 | 42 | $2.3 \cdot 10^{-8}$ | ⊘ |
| saylr4 | fixed | – | – | – | 12940 | 12940 | ⊘ | ⊘ | | ⊘ |
| | `us` | 13 | 6 | 2 | 45951 | 45950 | ⊘ | 278 | $6.8 \cdot 10^{-7}$ | ⊘ |
| | `ue` | 13 | 6 | 2 | 45951 | 45952 | ⊘ | 348 | $3.2 \cdot 10^{-7}$ | ⊘ |
| memplus | fixed | – | – | – | 70385 | 70069 | 789 | 366 | $1.6 \cdot 10^{-6}$ | ⊘ |
| | `us` | 5 | 2 | 2 | 84405 | 84405 | 536 | 169 | $1.9 \cdot 10^{-7}$ | ⊘ |
| | `ue` | 5 | 2 | 2 | 84405 | 84405 | 473 | 170 | $9.8 \cdot 10^{-8}$ | ⊘ |
| utm1700b | fixed | – | – | – | 11542 | 11603 | ⊘ | 836 | $6.4 \cdot 10^{-8}$ | ⊘ |
| | `us` | 13 | 6 | 2 | 19535 | 19367 | ⊘ | 410 | $7.0 \cdot 10^{-8}$ | ⊘ |
| | `ue` | 13 | 6 | 2 | 19302 | 19362 | ⊘ | 465 | $1.2 \cdot 10^{-7}$ | ⊘ |
| utm3060 | fixed | – | – | – | 22668 | 22541 | ⊘ | ⊘ | | ⊘ |
| | `us` | 17 | 8 | 2 | 48307 | 47549 | ⊘ | 371 | $1.1 \cdot 10^{-6}$ | ⊘ |
| | `ue` | 17 | 8 | 2 | 47444 | 47477 | ⊘ | 369 | $4.8 \cdot 10^{-7}$ | ⊘ |
| l_50_1 | fixed | – | – | – | 7400 | 7400 | 181 | 58 | $1.1 \cdot 10^{-8}$ | 613 |
| | `us` | 5 | 2 | 2 | 12347 | 12347 | 161 | 61 | $1.6 \cdot 10^{-8}$ | 411 |
| | `ue` | 5 | 2 | 2 | 12347 | 12347 | 146 | 53 | $5.2 \cdot 10^{-9}$ | 422 |
| l_50_100 | fixed | – | – | – | 7400 | 7400 | ⊘ | ⊘ | | ⊘ |
| | `us` | 9 | 4 | 2 | 22235 | 22236 | ⊘ | 69 | $2.1 \cdot 10^{-11}$ | ⊘ |
| | `ue` | 6 | 5 | 1 | 14985 | 14985 | 308 | 62 | $3.6 \cdot 10^{-11}$ | ⊘ |
| l_50_1000 | fixed | – | – | – | 7400 | 7400 | ⊘ | ⊘ | | ⊘ |
| | `us`, `ue` | | | | | | ⊘ | ⊘ | | ⊘ |
| c1m1o4 | fixed | – | – | – | 149388 | 149388 | 80 | 35 | $1.9 \cdot 10^{-8}$ | 218 |
| | `us` | 5 | 2 | 2 | 93983 | 105636 | 81 | 36 | $5.1 \cdot 10^{-9}$ | 226 |
| | `ue` | 5 | 2 | 2 | 93983 | 105215 | 225 | 39 | $7.5 \cdot 10^{-9}$ | 101 |
| c500m1o4 | fixed | – | – | – | 149388 | 149388 | ⊘ | ⊘ | | ⊘ |
| | `us`, `ue` | | | | | | ⊘ | ⊘ | | ⊘ |
| c100000m1o4 | fixed | – | – | – | 149388 | 149388 | ⊘ | ⊘ | | ⊘ |
| | `us`, `ue` | | | | | | ⊘ | ⊘ | | ⊘ |

Table 15: Performance of the solvers applied to the unsymmetric linear systems preconditioned by the `LU-projection` algorithm

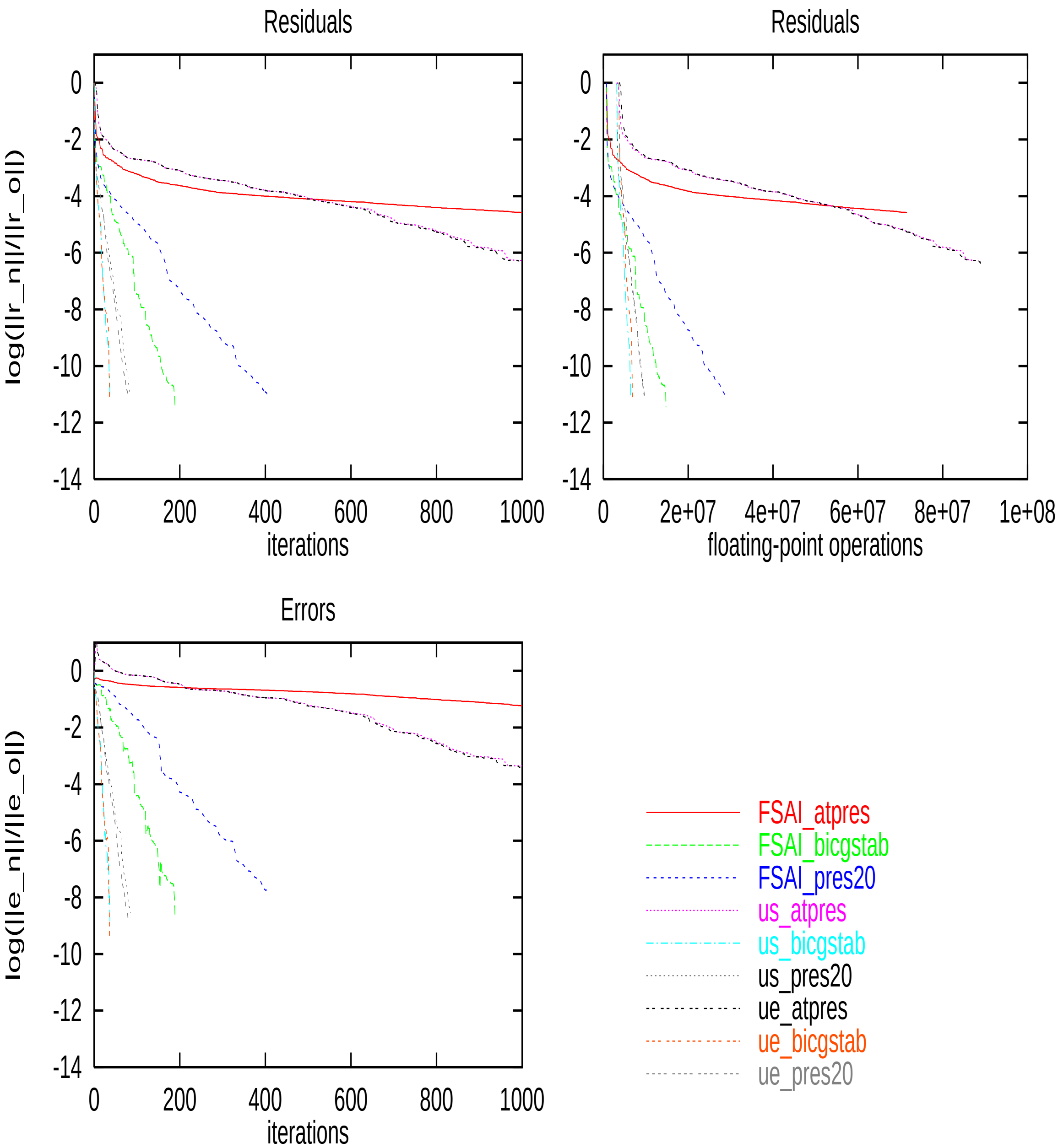


Figure 29: Convergence of the unsymmetric solvers preconditioned by the `LU-projection` method for the matrix orsirr2.

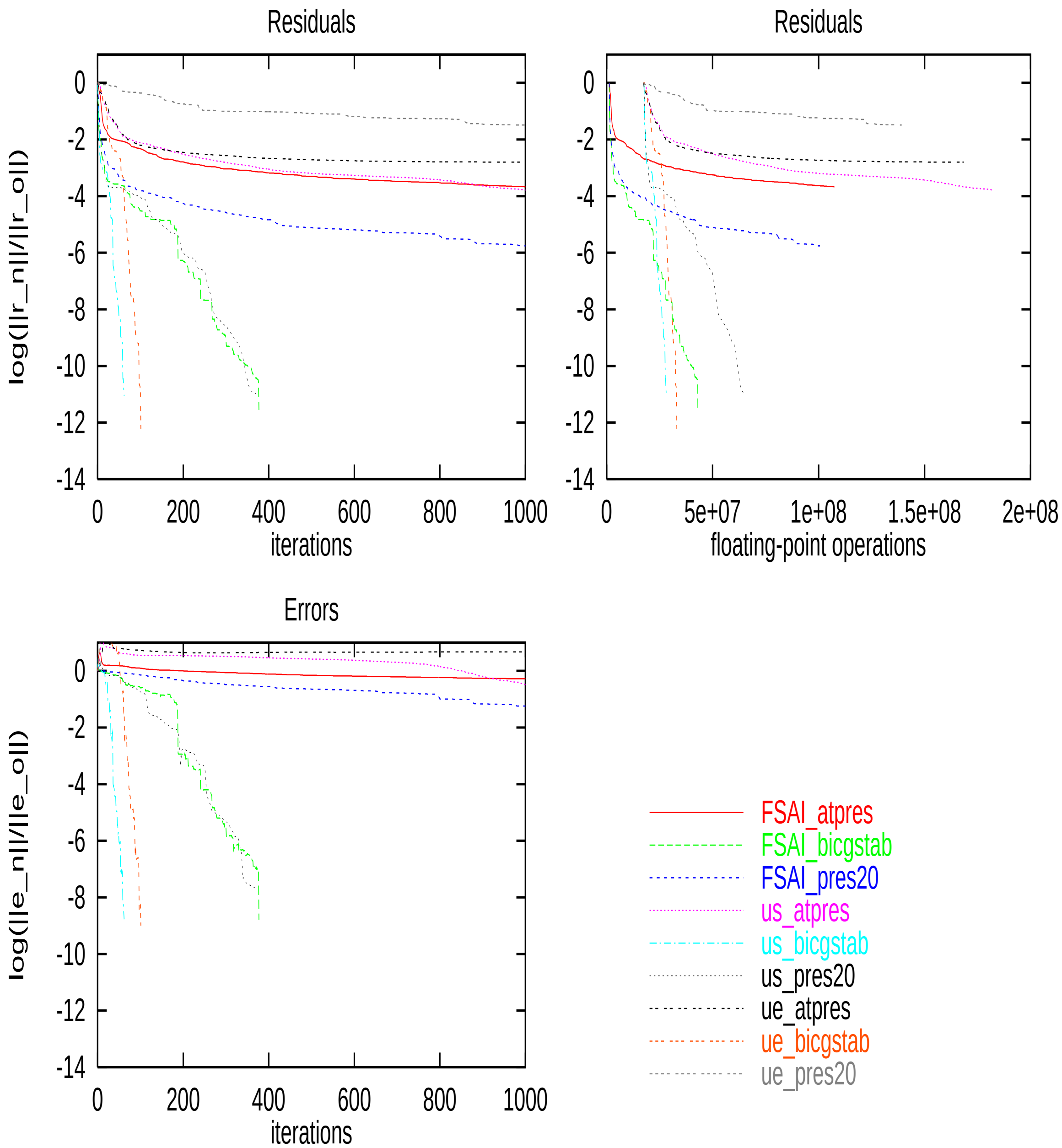


Figure 30: Convergence of the unsymmetric solvers preconditioned by the `LU-projection` method for the matrix `pores2`.

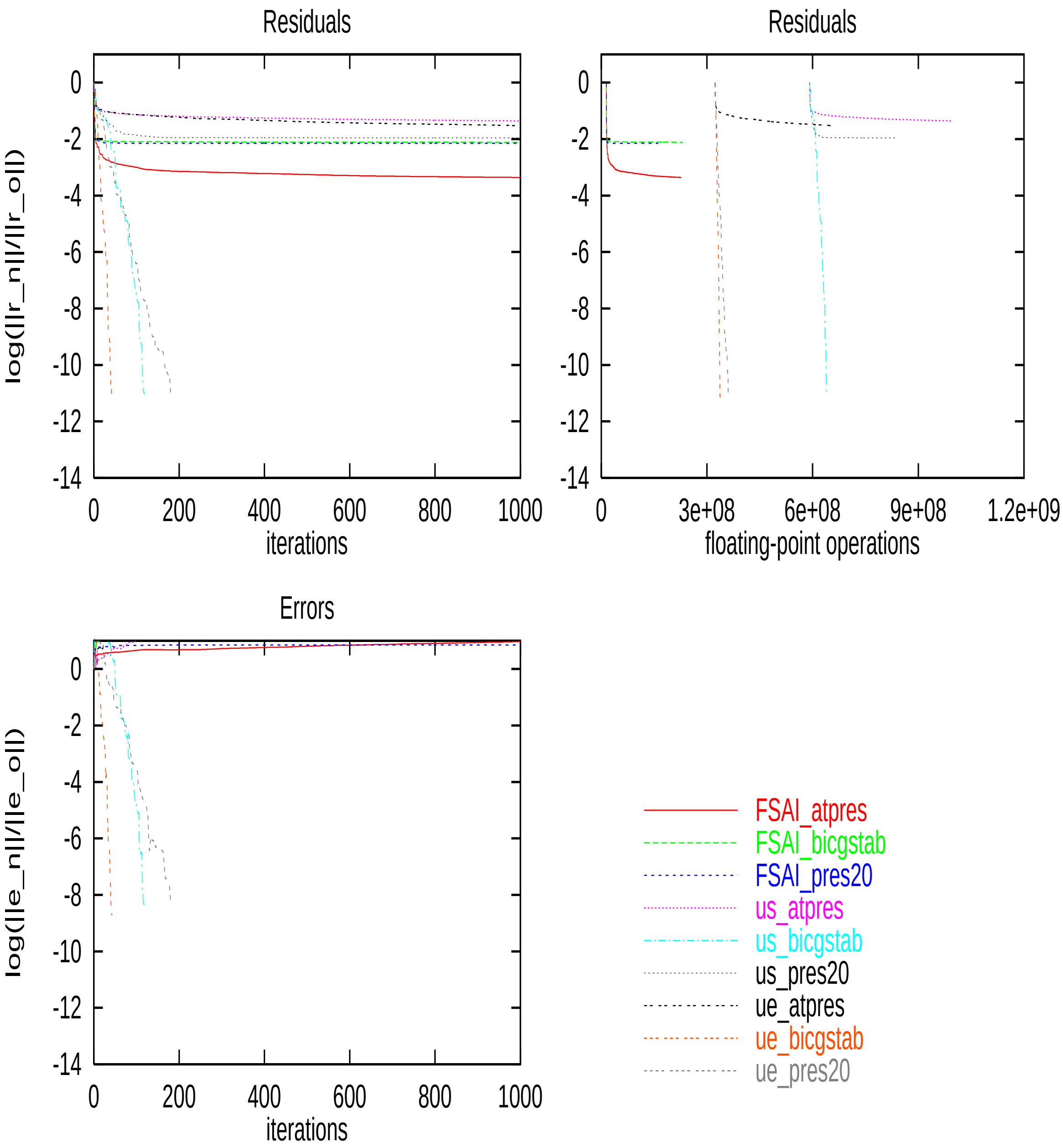


Figure 31: Convergence of the unsymmetric solvers preconditioned with the `LU-projection` method for the matrix `sherman2`.

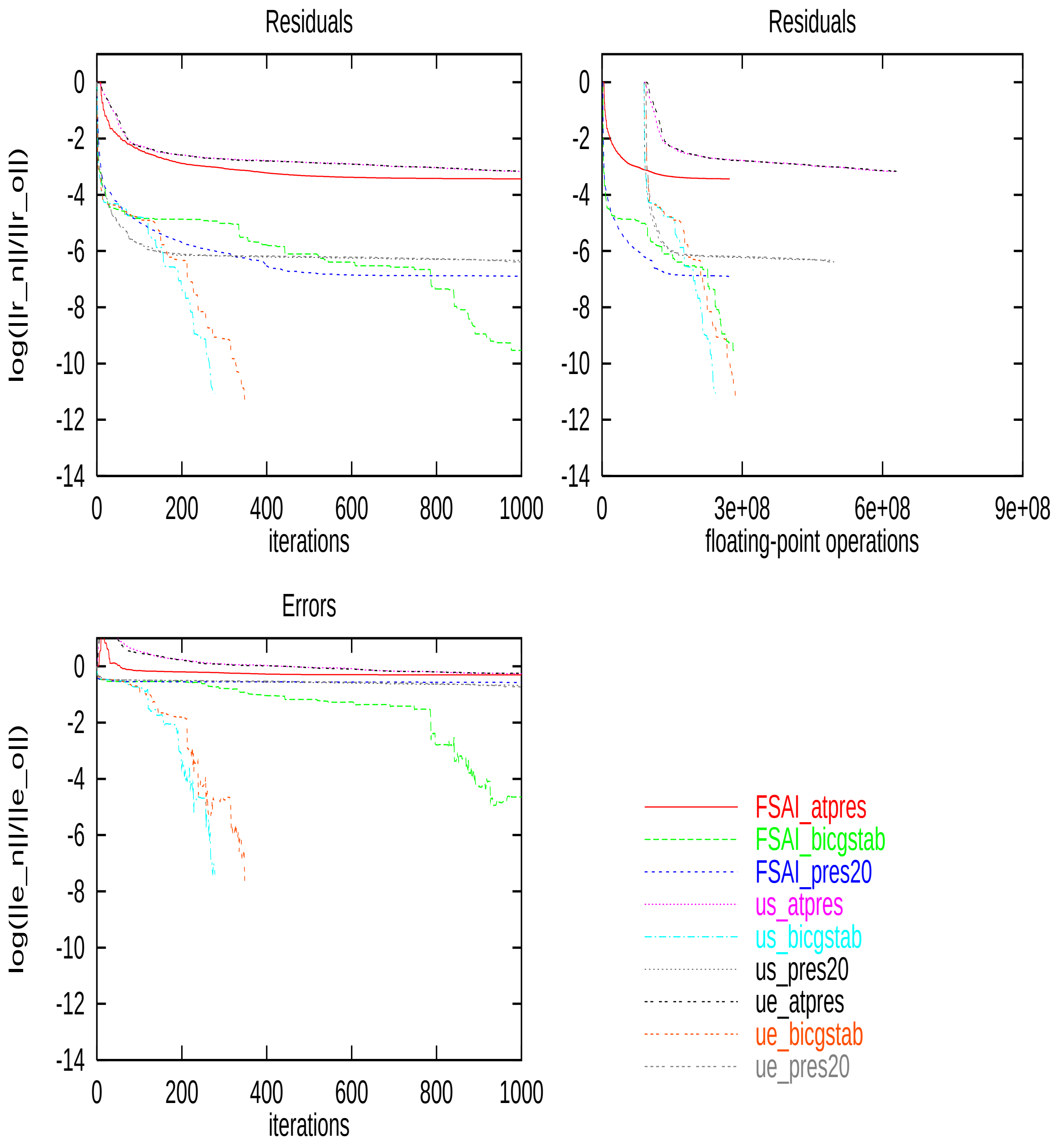


Figure 32: Convergence of the unsymmetric solvers preconditioned with the `LU-projection` method for the matrix saylr4.

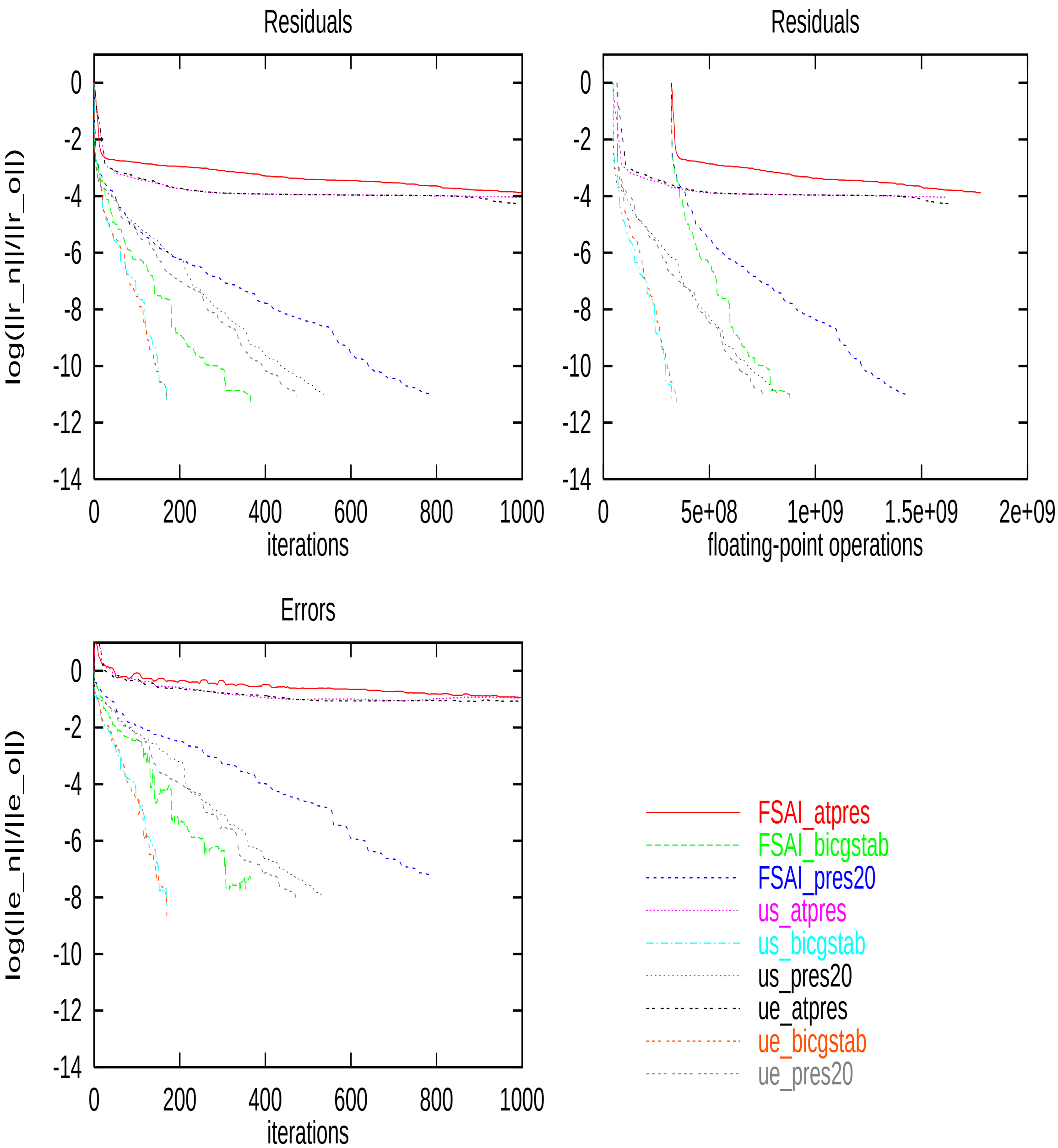


Figure 33: Convergence of the unsymmetric solvers preconditioned by the `LU-projection` method for the matrix memplus.

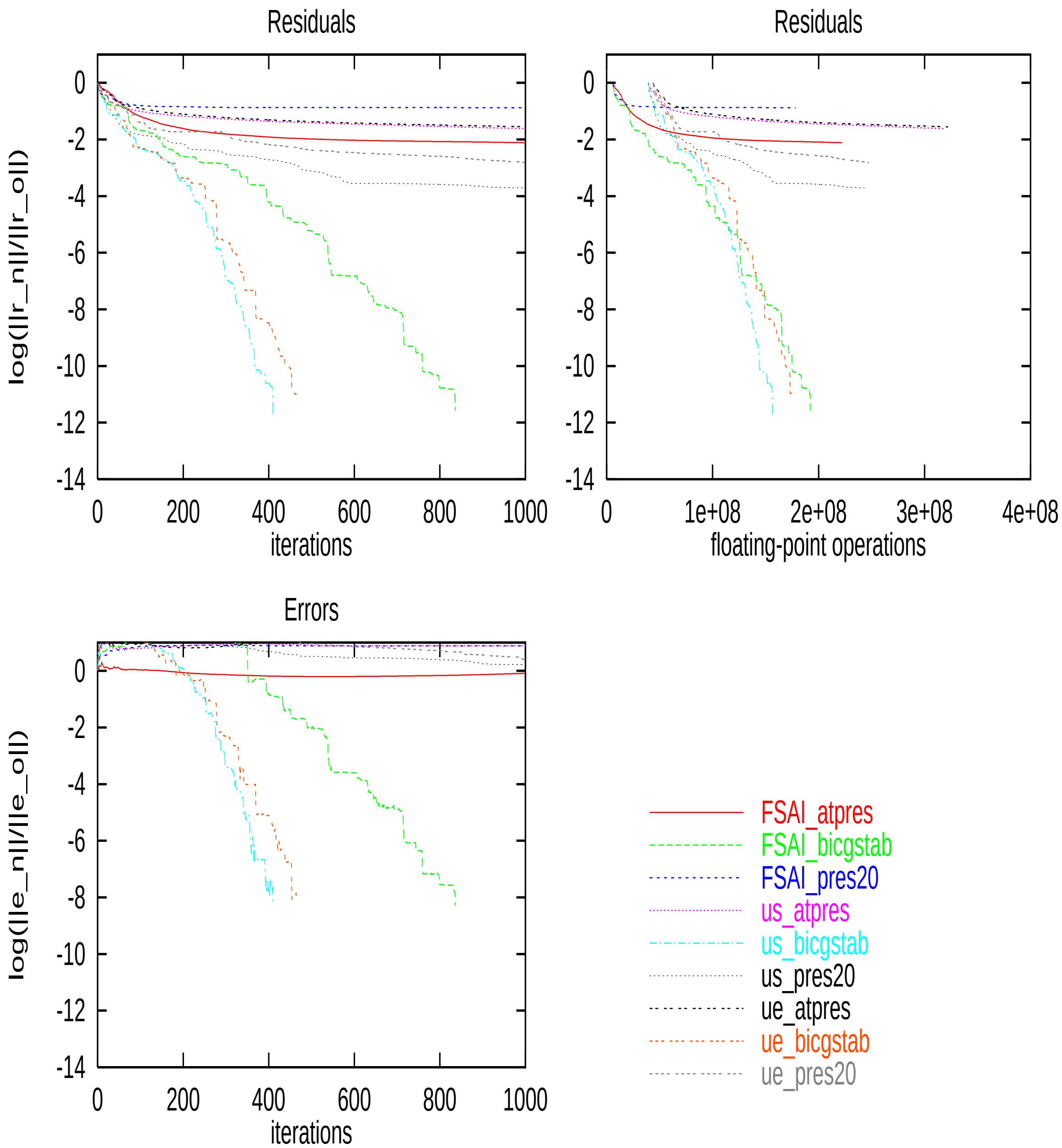


Figure 34: Convergence of the unsymmetric solvers preconditioned with the `LU-projection` method for the matrix `utm1700b`.

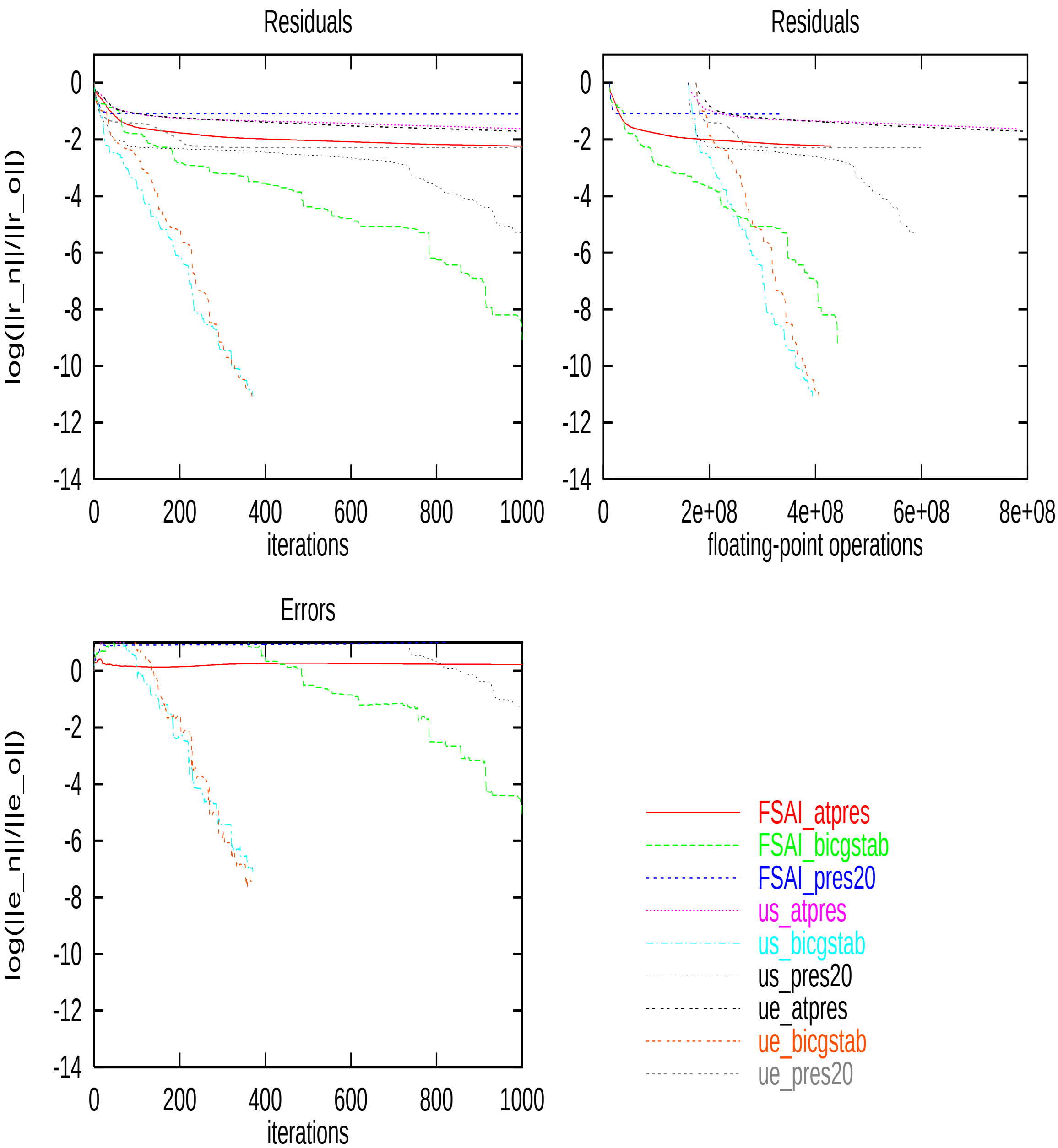


Figure 35: Convergence of the unsymmetric solvers preconditioned with the `LU-projection` method for the matrix `utm3060`.

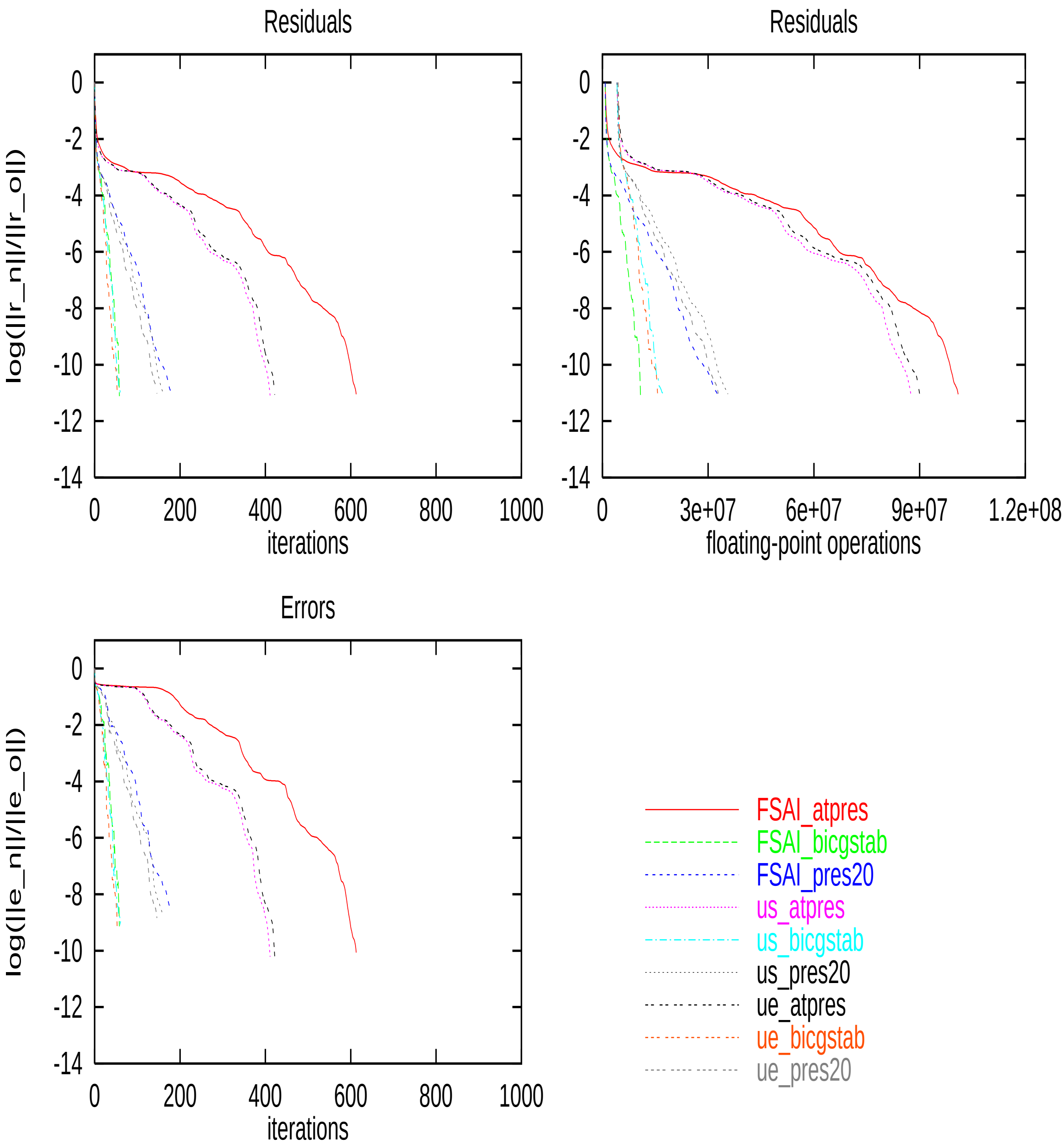


Figure 36: Convergence of the unsymmetric solvers preconditioned with the `LU-projection` method for the matrix l_50_1.

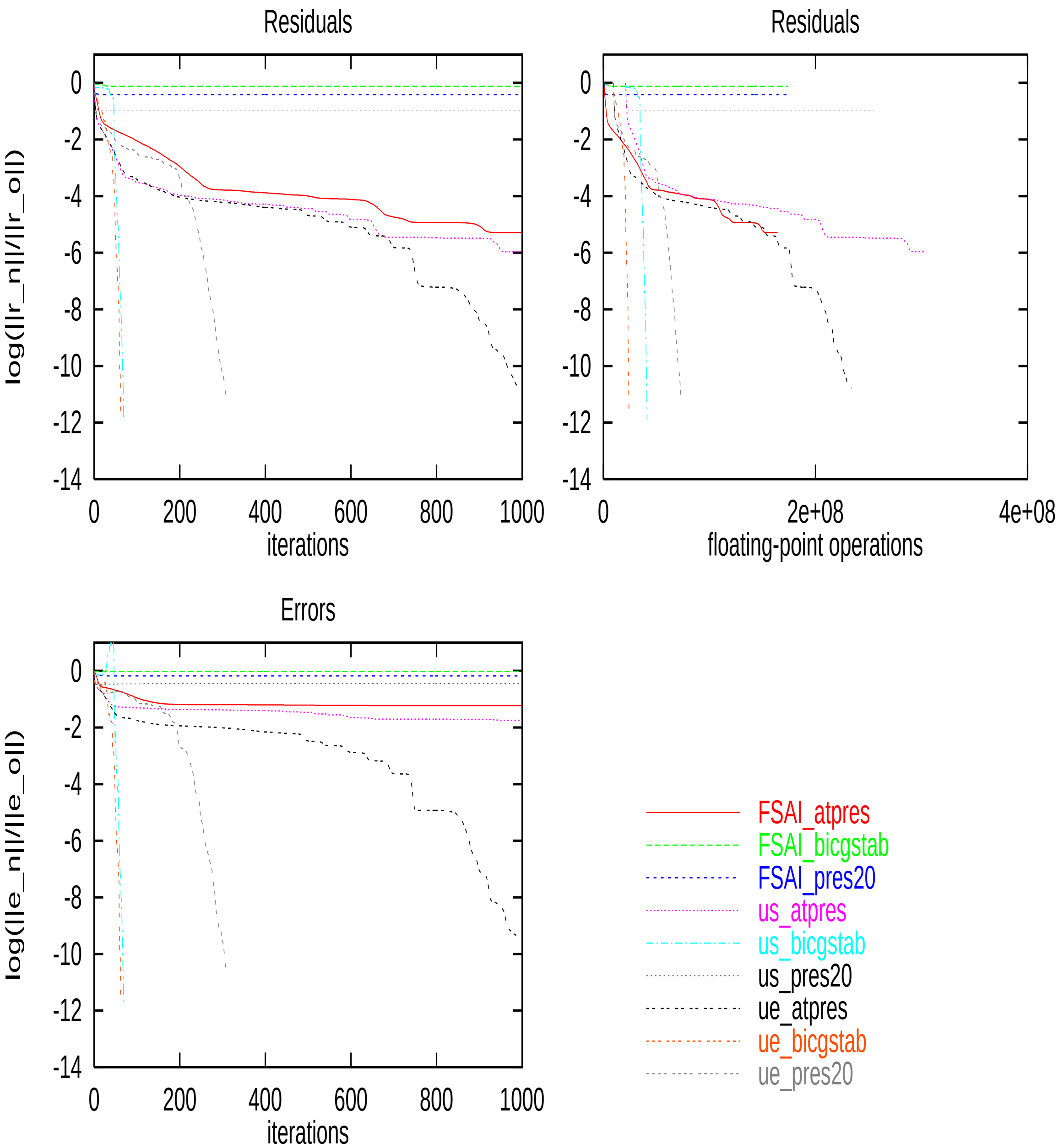


Figure 37: Convergence of the unsymmetric solvers preconditioned with the `LU-projection` method for the matrix l_50_100.

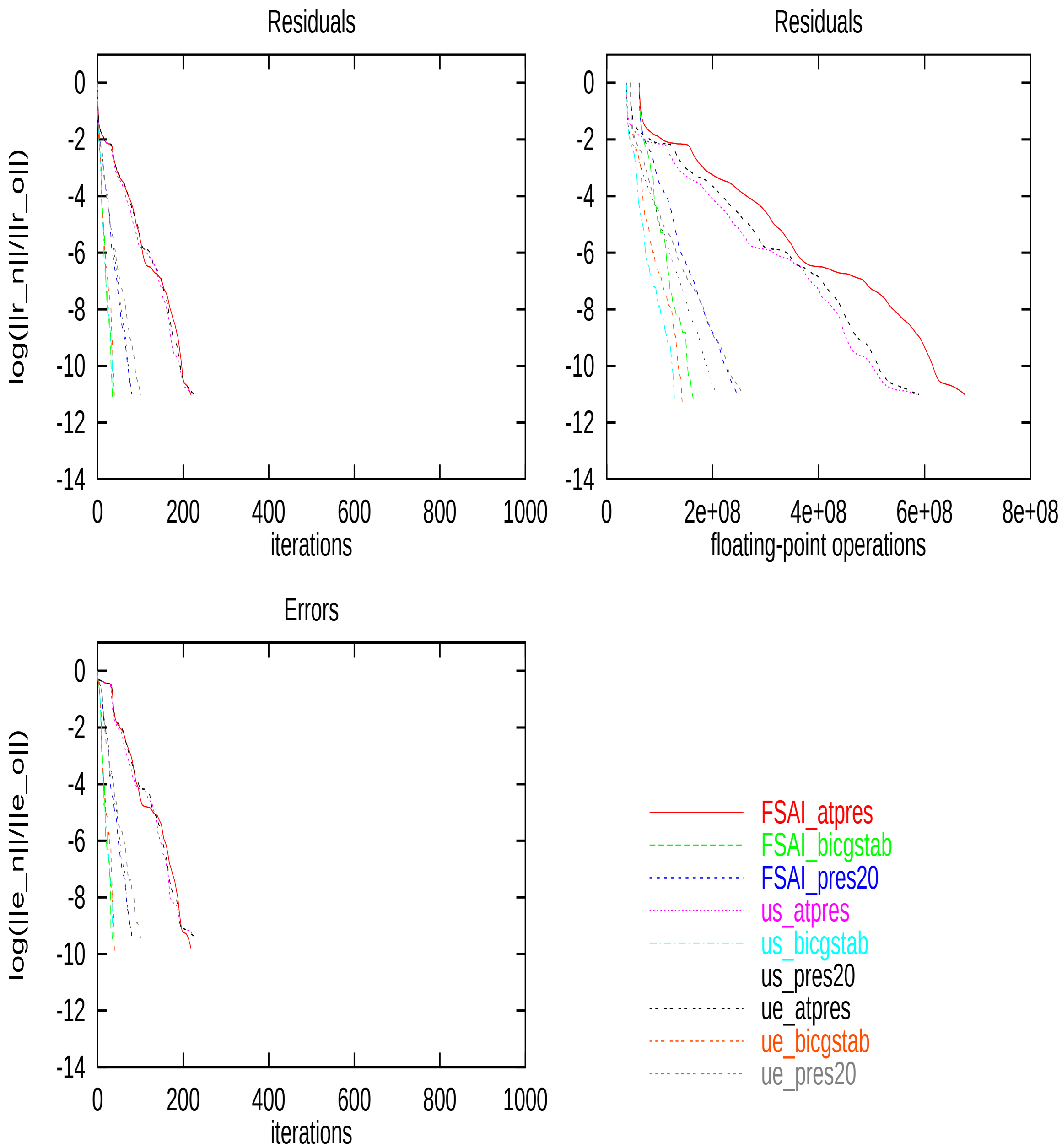


Figure 38: Convergence of the unsymmetric solvers preconditioned with the `LU-projection` method for the matrix `c1m1o4`.

### 7.5.3 Comparison of `Plain projection` and `LU-projection` to the Standard Preconditioning Techniques for Non-Symmetric Linear Systems

In this section we compare the performance of the `Plain projection` method and the `LU-projection` method to some state-of-the-art preconditioning techniques. The chosen methods are `ILU(0)` (see section 6.4), `SPAI` (see section 6.2) and `AINV` (see section 6.6).
Table 16 summarizes the results obtained by the `AINV` method. The three iterative solvers perform similar to the observations for `Plain projection` and `LU-projection`. While the `BiCGstab` iteration is by far the most stable iterative method (it fails only in two out of the 13 test cases), the `PRES20` iteration is much more unstable (it fails on seven test problems). The `ATPRES` iteration fails to converge in 10 out of the 13 tests.
The dropping threshold for the `AINV` method, as stated in the column denoted "thresh." in table 16, is chosen such that the obtained approximate inverses have at most twice the number of non-zeros, given in the column labeled "nz", of the original matrix. For the matrices utm1700b and utm3060, substantially more fill-in must be allowed to obtain a converging preconditioned iteration. The dropping thresholds are obtained by adjusting them along a few test runs.

| matrix | thresh. | nz | PRES20 | BiCGstab | | ATPRES |
|---|---|---|---|---|---|---|
| | | | its | its | $\|\|x^* - x_k\|\|_2$ | its |
| orsirr2 | 0.1 | 4903 | 76 | 41 | $1.9 \cdot 10^{-8}$ | $\oslash$ |
| pores2 | 0.1 | 12473 | $\oslash$ | 54 | $9.8 \cdot 10^{-9}$ | $\oslash$ |
| sherman2 | 0.01 | 23462 | 476 | 223 | $1.6 \cdot 10^{-9}$ | $\oslash$ |
| saylr4 | 0.1 | 49231 | $\oslash$ | 53 | $1.5 \cdot 10^{-7}$ | $\oslash$ |
| memplus | 0.05 | 68142 | 346 | 229 | $9.6 \cdot 10^{-8}$ | $\oslash$ |
| utm1700b | 0.1 | 135191 | $\oslash$ | 300 | $5.8 \cdot 10^{-5}$ | $\oslash$ |
| utm3060 | 0.1 | 274862 | $\oslash$ | 249 | $1.1 \cdot 10^{-4}$ | $\oslash$ |
| l_50_1 | 0.1 | 12540 | 204 | 64 | $1.2 \cdot 10^{-8}$ | 614 |
| l_50_100 | 0.1 | 70191 | 43 | 16 | $4.7 \cdot 10^{-11}$ | 94 |
| l_50_1000 | | | $\oslash$ | $\oslash$ | | $\oslash$ |
| c1m1o4 | 0.1 | 123162 | 86 | 36 | $3.1 \cdot 10^{-9}$ | 231 |
| c500m1o4 | 0.1 | 397967 | $\oslash$ | 181 | $2.4 \cdot 10^{-8}$ | $\oslash$ |
| c100000m1o4 | | | $\oslash$ | $\oslash$ | | $\oslash$ |

Table 16: Performance of the solvers applied to the unsymmetric linear systems preconditioned with the `AINV` method

The results for the `ILU(0)` preconditioning technique are summarized in table 17. Further, for comparison, in table 17 the performance of the unpreconditioned iterations, as given in tables 3, 5 and 9, is included. The `BiCGstab` iteration with `ILU(0)` preconditioning converges in 11 of the 13 test cases; it fails to converge for the matrices l_50_1000 and c100000m1o4 only. As indicated by the ‡-symbol, for the matrix l_50_1000, the iterations with `ILU(0)` preconditioning were stopped due to a relative deviation of the updated residuals to the original ones of more than $10^{-5}$ in the Euclidian norm (see page 155).

| matrix | ILU(0) | | | | unpreconditioned | | | |
|---|---|---|---|---|---|---|---|---|
| | PRES20 | BiCGstab | | ATPRES | PRES20 | BiCGstab | | ATPRES |
| | its | its | $\|\|x^* - x_k\|\|_2$ | its | its | its | $\|\|x^* - x_k\|\|_2$ | its |
| orsirr2 | 66 | 34 | $1.3 \cdot 10^{-8}$ | 591 | 668 | 367 | $1.1 \cdot 10^{-7}$ | ⊘ |
| pores2 | 297 | 31 | $7.8 \cdot 10^{-9}$ | ⊘ | ⊘ | ⊘ | | ⊘ |
| sherman2 | 22 | 10 | $8.6 \cdot 10^{-11}$ | ⊘ | ⊘ | ⊘ | | ⊘ |
| saylr4 | ⊘ | 45 | $7.4 \cdot 10^{-8}$ | ⊘ | ⊘ | ⊘ | | ⊘ |
| memplus | 851 | 360 | $8.3 \cdot 10^{-7}$ | ⊘ | ⊘ | 765 | $3.9 \cdot 10^{-6}$ | ⊘ |
| utm1700b | 818 | 101 | $2.4 \cdot 10^{-8}$ | ⊘ | ⊘ | ⊘ | | ⊘ |
| utm3060 | 804 | 104 | $1.5 \cdot 10^{-7}$ | ⊘ | ⊘ | ⊘ | | ⊘ |
| l_50_1 | 111 | 37 | $2.4 \cdot 10^{-9}$ | 228 | 496 | 118 | $5.2 \cdot 10^{-9}$ | ⊘ |
| l_50_100 | 292 | 18 | $6.0 \cdot 10^{-7}$ | ⊘ | 266 | 215 | $5.8 \cdot 10^{-11}$ | 424 |
| l_50_1000 | ‡ | ‡ | | ‡ | 531 | ⊘ | | 683 |
| c1m1o4 | 49 | 27 | $3.8 \cdot 10^{-9}$ | 120 | 193 | 72 | $3.9 \cdot 10^{-9}$ | 707 |
| c500m1o4 | 212 | 87 | $2.1 \cdot 10^{-9}$ | ⊘ | ⊘ | ⊘ | | ⊘ |
| c100000m1o4 | ⊘ | ⊘ | | ⊘ | ⊘ | ⊘ | | ⊘ |

Table 17: Performance of the solvers applied to the unsymmetric linear systems preconditioned with the `ILU(0)` method

The results obtained for the `SPAI` preconditioning technique are summarized in table 18. Similar to the previous tests, the differences in the performance of the iterative solvers are dramatic. While the `BiCGstab` iteration fails only in a few cases, the other two iterations have a high failure rate.
The column "decrease rates" in table 18 denotes the two variants of the `SPAI` algorithm obtained by utilizing either the univariate decrease rates defined in (6.17) (these rows have the entry "`us`") or the decrease rates defined in (6.18) (the corresponding rows have the entry "`m`"). In all tests, the computational complexity of the `SPAI` variants is dominated by the pattern derivation. Because the computational cost for both variants of the `SPAI` algorithm greatly exceeds the computational cost

for all other preconditioning techniques, numerical dropping in the residual, as discussed on page 157, is applied.
We consider the performance of the `BiCGstab` iteration: for four matrices of the test set (these are utm1700b, utm3060, l_50_1000 and c100000m1o4), no parameters for the `SPAI` algorithm, yielding a converging preconditioned iteration, could be found. Apart from the higher computational cost for the variant using the decrease rates from (6.18) for the pattern derivation, both variants of the `SPAI` algorithm perform about the same.

| matrix | decrease | `SPAI` parameters | | | | nz$(P)$ | `PRES20` | `BiCGstab` | | `ATPRES` |
|---|---|---|---|---|---|---|---|---|---|---|
| | rates | $mf$ | $ms$ | $mfps$ | $\epsilon_k$ | | its | its | $\|\|x^* - x_k\|\|_2$ | its |
| orsirr2 | `us` | 11 | 5 | 2 | 0.5 | 3257 | 226 | 71 | $1.9 \cdot 10^{-8}$ | 545 |
| | `m` | 11 | 5 | 2 | 0.5 | 3174 | 191 | 63 | $1.5 \cdot 10^{-8}$ | 539 |
| pores2 | `us` | 41 | 4 | 10 | 0.3 | 16932 | ⊘ | 153 | $4.5 \cdot 10^{-8}$ | 957 |
| | `m` | 41 | 4 | 10 | 0.3 | 16882 | ⊘ | 132 | $2.3 \cdot 10^{-8}$ | 882 |
| sherman2 | `us` | 51 | 10 | 5 | 0.3 | 11588 | 65 | 33 | $1.8 \cdot 10^{-9}$ | 107 |
| | `m` | 51 | 10 | 5 | 0.3 | 11098 | 26 | 14 | $1.2 \cdot 10^{-8}$ | 55 |
| saylr4 | `us` | 51 | 10 | 5 | 0.3 | 38633 | ⊘ | 274 | $1.1 \cdot 10^{-6}$ | ⊘ |
| | `m` | 51 | 10 | 5 | 0.3 | 38184 | ⊘ | 241 | $1.1 \cdot 10^{-5}$ | ⊘ |
| memplus | `us` | 16 | 3 | 5 | 0.5 | 43933 | 810 | 378 | $1.1 \cdot 10^{-6}$ | 880 |
| | `m` | 16 | 3 | 5 | 0.5 | 43933 | 536 | 470 | $1.3 \cdot 10^{-6}$ | 853 |
| utm1700b | `us,m` | | | | | | ⊘ | ⊘ | | ⊘ |
| utm3060 | `us,m` | | | | | | ⊘ | ⊘ | | ⊘ |
| l_50_1 | `us` | 5 | 2 | 2 | 0.4 | 7126 | 413 | 108 | $5.2 \cdot 10^{-9}$ | ⊘ |
| | `m` | 7 | 3 | 2 | 0.4 | 7126 | 413 | 108 | $5.1 \cdot 10^{-9}$ | ⊘ |
| l_50_100 | `us` | 13 | 3 | 4 | 0.4 | 22092 | 183 | 56 | $2.6 \cdot 10^{-10}$ | 582 |
| | `m` | 13 | 3 | 4 | 0.4 | 22092 | 183 | 58 | $7.3 \cdot 10^{-10}$ | 582 |
| l_50_1000 | `us,m` | | | | | | ⊘ | ⊘ | | ⊘ |
| c1m1o4 | `us` | 7 | 1 | 6 | 0.4 | 43596 | 148 | 74 | $1.1 \cdot 10^{-9}$ | 702 |
| | `us` | 7 | 1 | 6 | 0.4 | 43596 | 155 | 66 | $9.2 \cdot 10^{-9}$ | 703 |
| c500m1o4 | `us` | 13 | 2 | 6 | 0.4 | 129984 | ⊘ | 203 | $8.9 \cdot 10^{-8}$ | ⊘ |
| | `m` | 13 | 2 | 6 | 0.4 | 129744 | ⊘ | 201 | $4.7 \cdot 10^{-9}$ | ⊘ |
| c100000m1o4 | `us,m` | | | | | | ⊘ | ⊘ | | ⊘ |

Table 18: Performance of the solvers applied to the unsymmetric linear systems preconditioned with the `SPAI` method

**Comparison of the Preconditioners for the Non-Symmetric Linear Systems**

In figures 39–50, the plots of the `BiCGstab` iterations without preconditioning and preconditioned by `SPAI`, `FSAI`, `LU-projection`, `Plain projection`, `ILU(0)` and `AINV` are shown (the terms "us", "ue" and "m" in these figures refer to the utilized decrease rates for `SPAI` (see page 217), for `Plain projection` (see page 186) and for `LU-projection` (see page 203).

The robustness of `ILU(0)`, `AINV` and `Plain projection`, regarding the number of not converging `BiCGstab` iterations, is comparable. All these methods fail for the matrices l_50_1000 and c100000m1o4 only. The `SPAI` and `LU-projection` preconditioners fail on slightly more problems. The `FSAI` method is not stable; it fails on seven out of the 13 test problems, and in case of convergence the acceleration of the iteration is slow.

As for the computational complexity of the tested methods, `ILU(0)` and `AINV` are the cheapest methods. The pattern adaptive projection methods `Plain projection` and `LU-projection` are more expensive, at least in a sequential environment. The computational complexity for constructing the `SPAI` preconditioners exceeds the cost for the other methods by far.

We summarize the results obtained for the considered preconditioning techniques for the `BiCGstab` solver in table 19:

| matrix | unpre-conditioned | `FSAI` | `SPAI` (us) | `LU-projection` (us) | `Plain projection`(us) | `ILU(0)` | `AINV` |
|---|---|---|---|---|---|---|---|
| orsirr2 | 367 | 189 | 71 | **37** | $\boxed{51}$ | $\boxed{\mathbf{34}}$ | $\boxed{\mathbf{41}}$ |
| pores2 | ⊘ | 377 | 153 | **62** | $\boxed{128}$ | $\boxed{\mathbf{31}}$ | $\boxed{\mathbf{54}}$ |
| sherman2 | ⊘ | ⊘ | **33** | 119 | $\boxed{\mathbf{35}}$ | $\boxed{\mathbf{10}}$ | $\boxed{223}$ |
| saylr4 | ⊘ | ⊘ | **274** | 278 | $\boxed{354}$ | $\boxed{\mathbf{45}}$ | $\boxed{\mathbf{53}}$ |
| memplus | 765 | 366 | 378 | **169** | $\boxed{372}$ | $\boxed{\mathbf{360}}$ | $\boxed{\mathbf{229}}$ |
| utm1700b | ⊘ | 836 | ⊘ | $\boxed{410}$ | $\boxed{\mathbf{358}}$ | $\boxed{\mathbf{101}}$ | **300** |
| utm3060 | ⊘ | ⊘ | ⊘ | **371** | $\boxed{401}$ | $\boxed{\mathbf{104}}$ | $\boxed{\mathbf{249}}$ |
| l_50_1 | 118 | $\boxed{\mathbf{58}}$ | 108 | **61** | 73 | $\boxed{\mathbf{37}}$ | $\boxed{64}$ |
| l_50_100 | $\boxed{215}$ | ⊘ | 56 | 69 | **53** | $\boxed{\mathbf{18}}$ | $\boxed{\mathbf{16}}$ |
| l_50_1000 | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ‡ | ⊘ |
| c1m1o4 | 72 | **35** | 74 | **36** | $\boxed{61}$ | $\boxed{\mathbf{27}}$ | $\boxed{\mathbf{36}}$ |
| c500m1o4 | ⊘ | ⊘ | **203** | ⊘ | $\boxed{264}$ | $\boxed{\mathbf{87}}$ | $\boxed{\mathbf{181}}$ |
| c100000m1o4 | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ |

Table 19: Comparison of the preconditioners for the unsymmetric test problems for the `BiCGstab` iteration

In table 19, the three iterations with the fewest iteration steps are written in boldface. Further, the three iterations with the smallest overall number of floating-point operations (this is set-up cost for the preconditioner plus cost for the preconditioned iteration) are contained in oval boxes. With this notation, the efficiency and the robustness of `ILU(0)` and `AINV` becomes obvious. The newly proposed preconditioning techniques `Plain projection` and `LU-projection` perform comparable to `ILU(0)` and `AINV`, although the set-up cost for `Plain projection` and `LU-projection` is, in a sequential environment, larger than for `ILU(0)` and `AINV`. However, since `Plain projection` and `LU-projection` are inherently parallel, in a parallel environment this situation may be reversed.

With the results displayed in table 19, it appears that the `FSAI` method is not an efficient preconditioning technique. Further, because `FSAI` works without adaptive pattern derivation, it is clearly inferior to the `LU-projection` method.

The `SPAI` algorithm is, although in robustness comparable to `LU-projection` and `Plain projection`, due to its enormous set-up cost inferior to `LU-projection` and `Plain projection`.

None of the tested preconditioners worked sufficiently for the matrices l_50_1000 and `c100000m1o4`. Interestingly, for the matrix l_50_1000, the unpreconditioned `PRES20` and `ATPRES` solvers converge (see table 5 and figure 48). Further, if the `Plain projection` algorithm is applied without adaptive pattern derivation, i.e. if the pattern of the matrix l_50_1000 is used as the fixed projection pattern, the `Plain projection` determines a projective approximate inverse which converges with the `BiCGstab` iteration (see table 14 and figure 48). This shows, that sparse approximate inverse preconditioning for the matrix l_50_1000 is possible. However, analogous results do not hold for the matrix `c100000m1o4`.

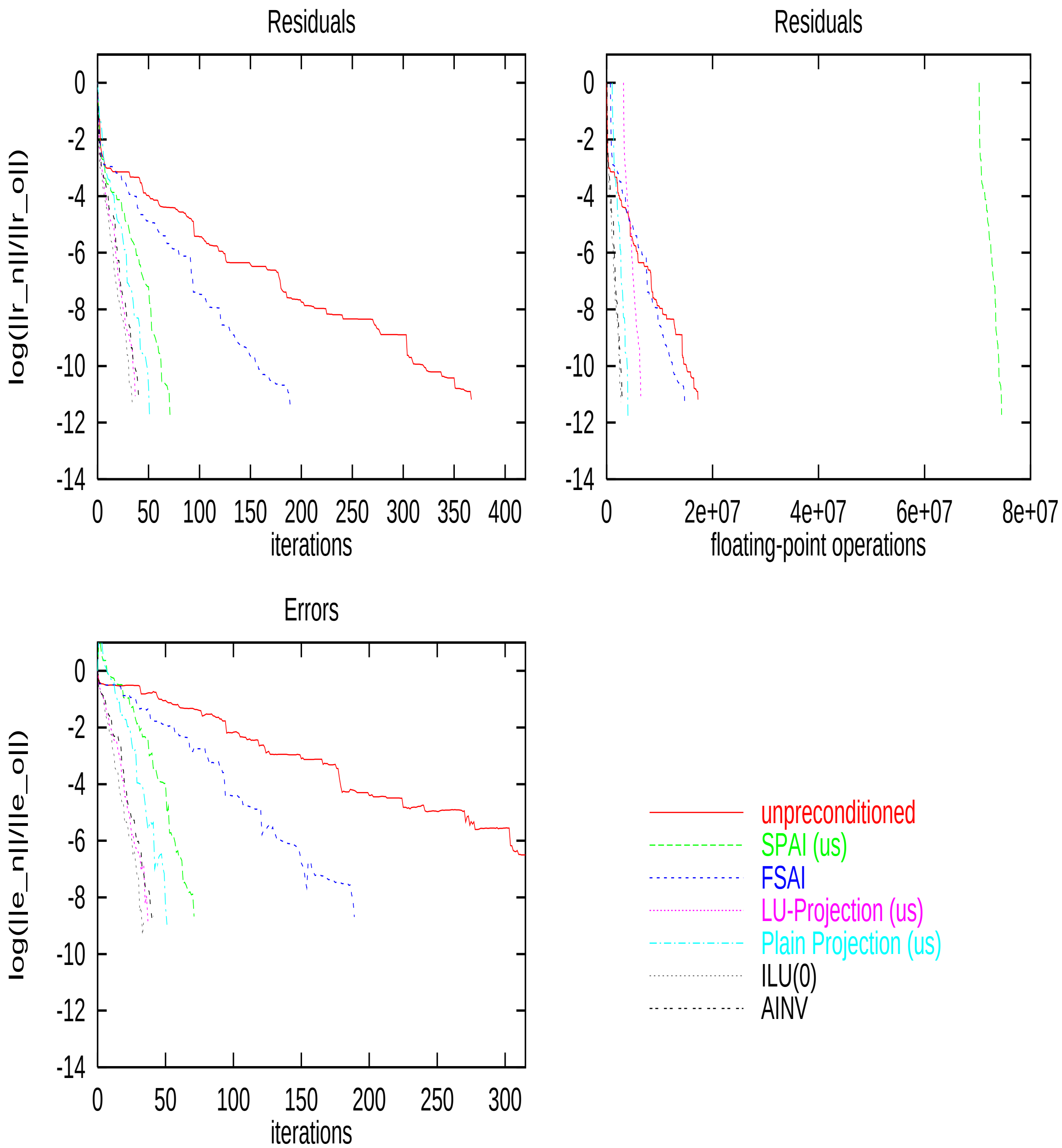


Figure 39: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix orsirr2.

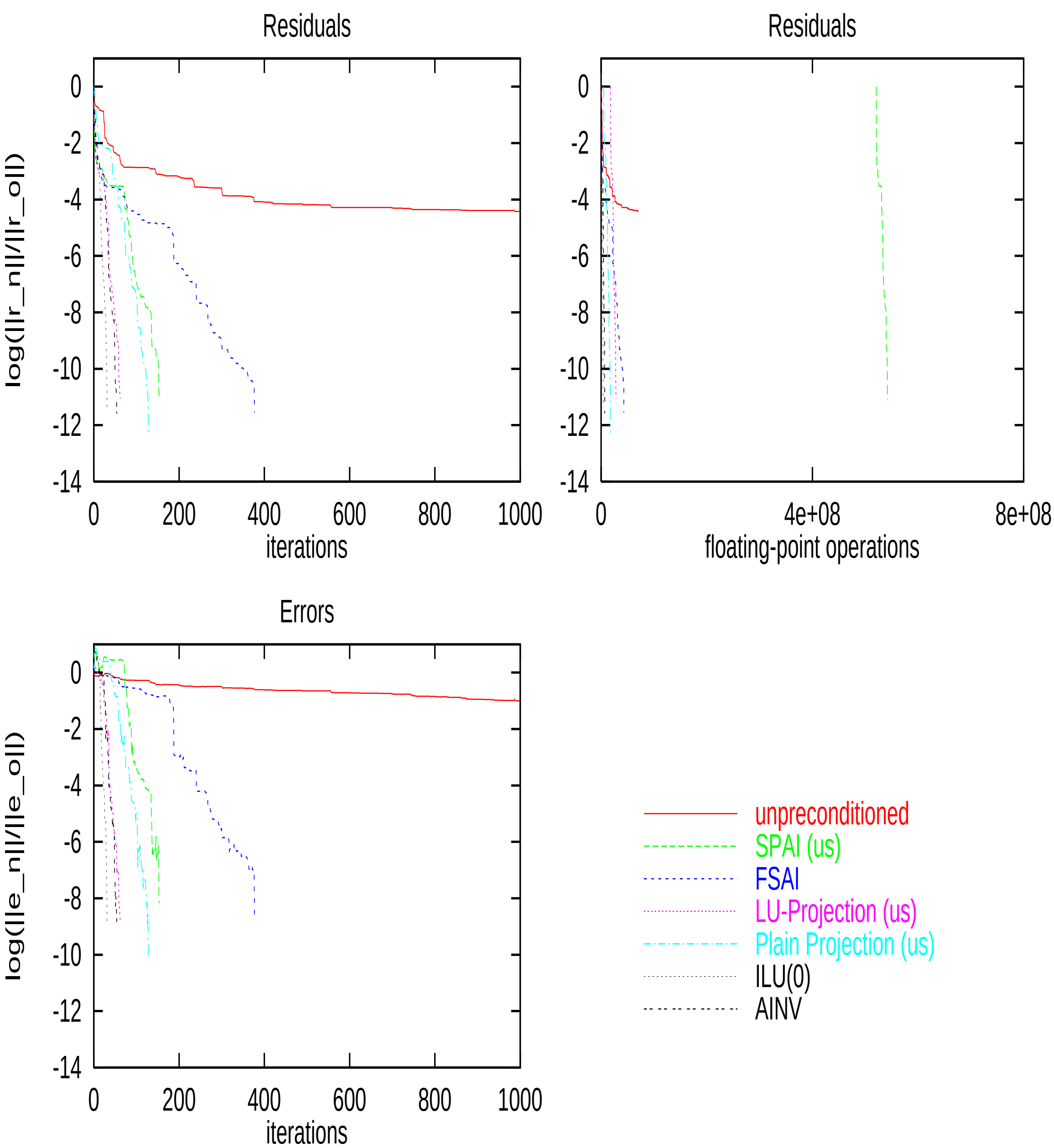


Figure 40: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix `pores2`.

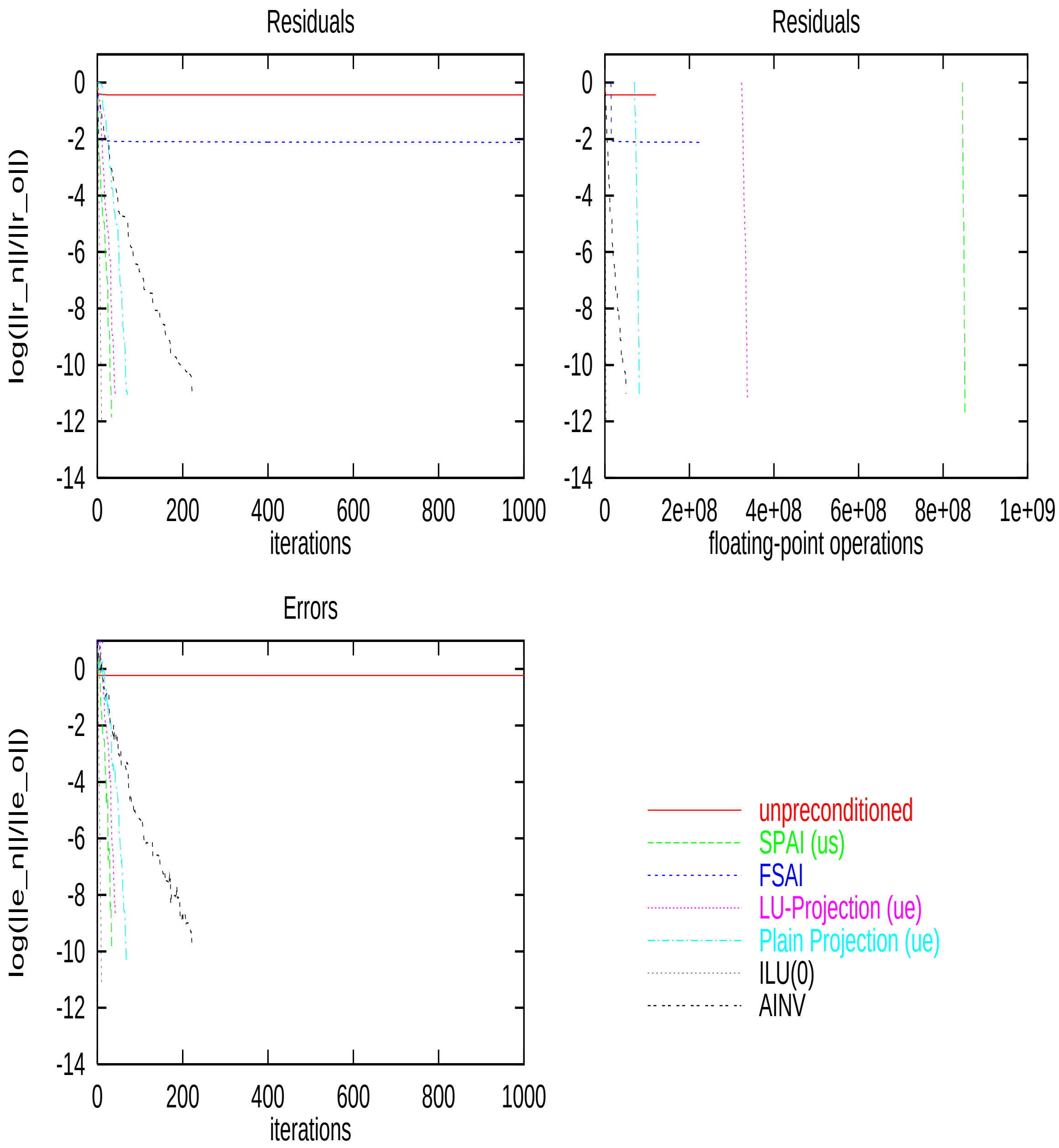


Figure 41: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix `sherman2`.

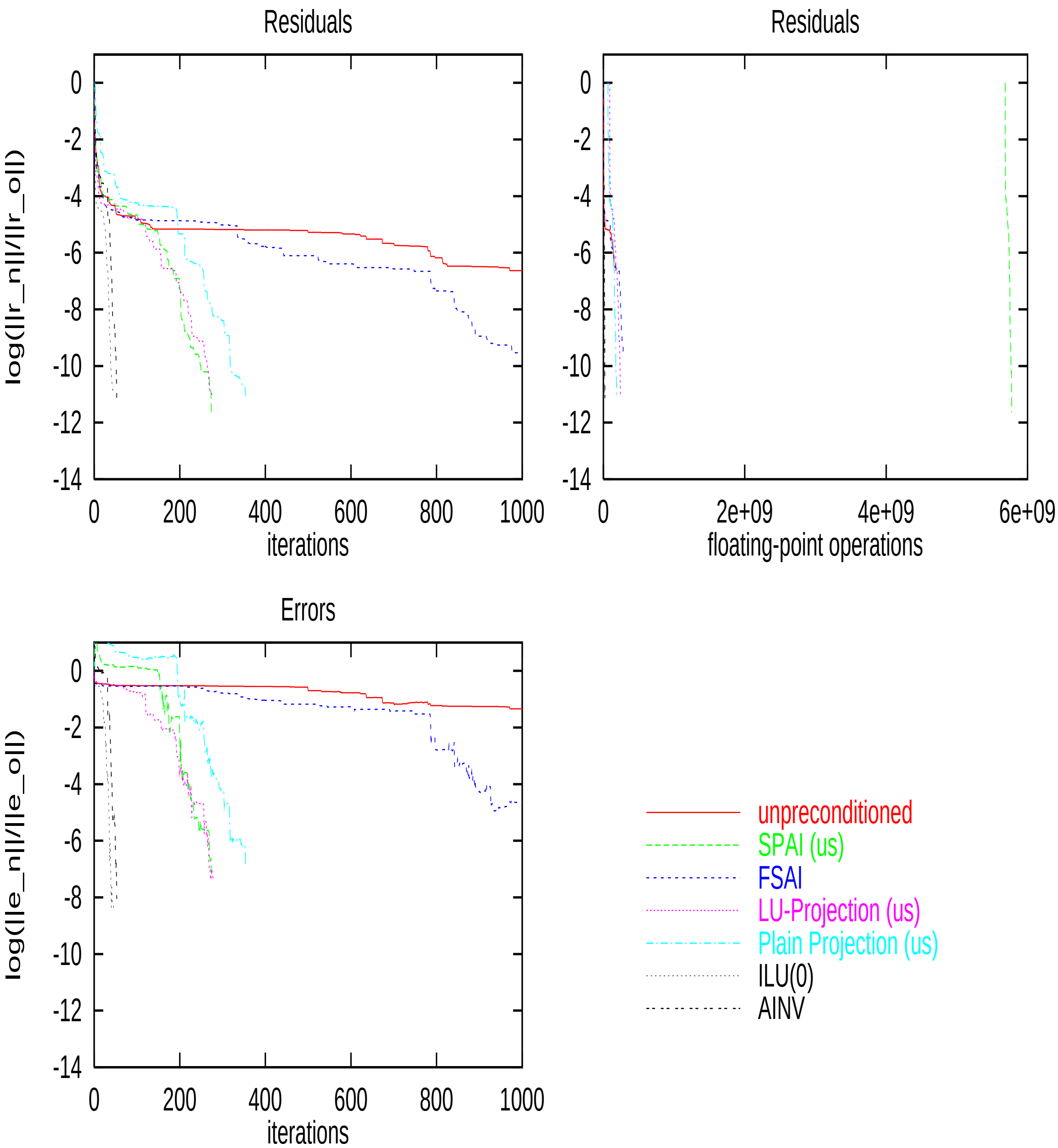


Figure 42: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix saylr4.

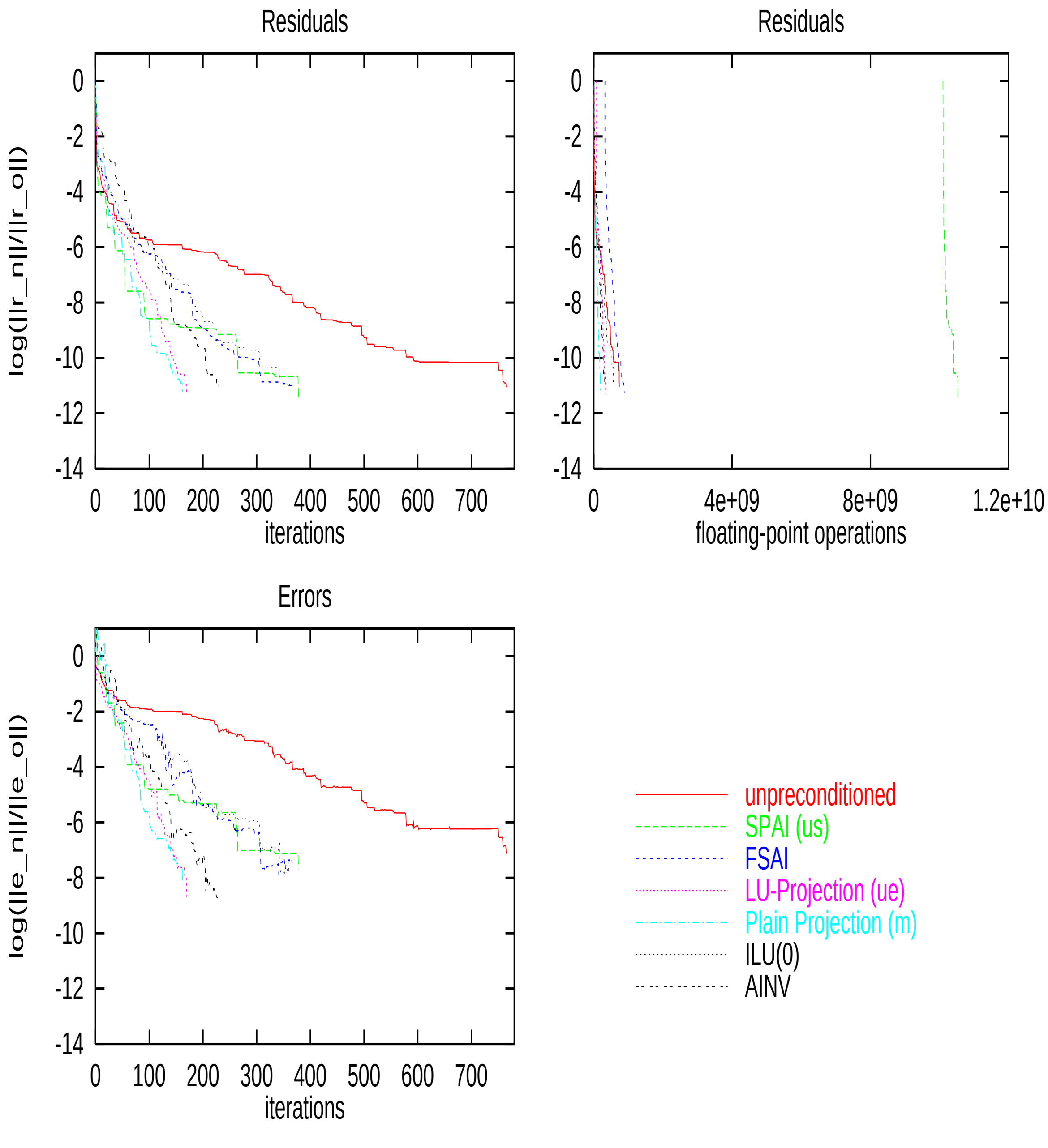


Figure 43: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix memplus.

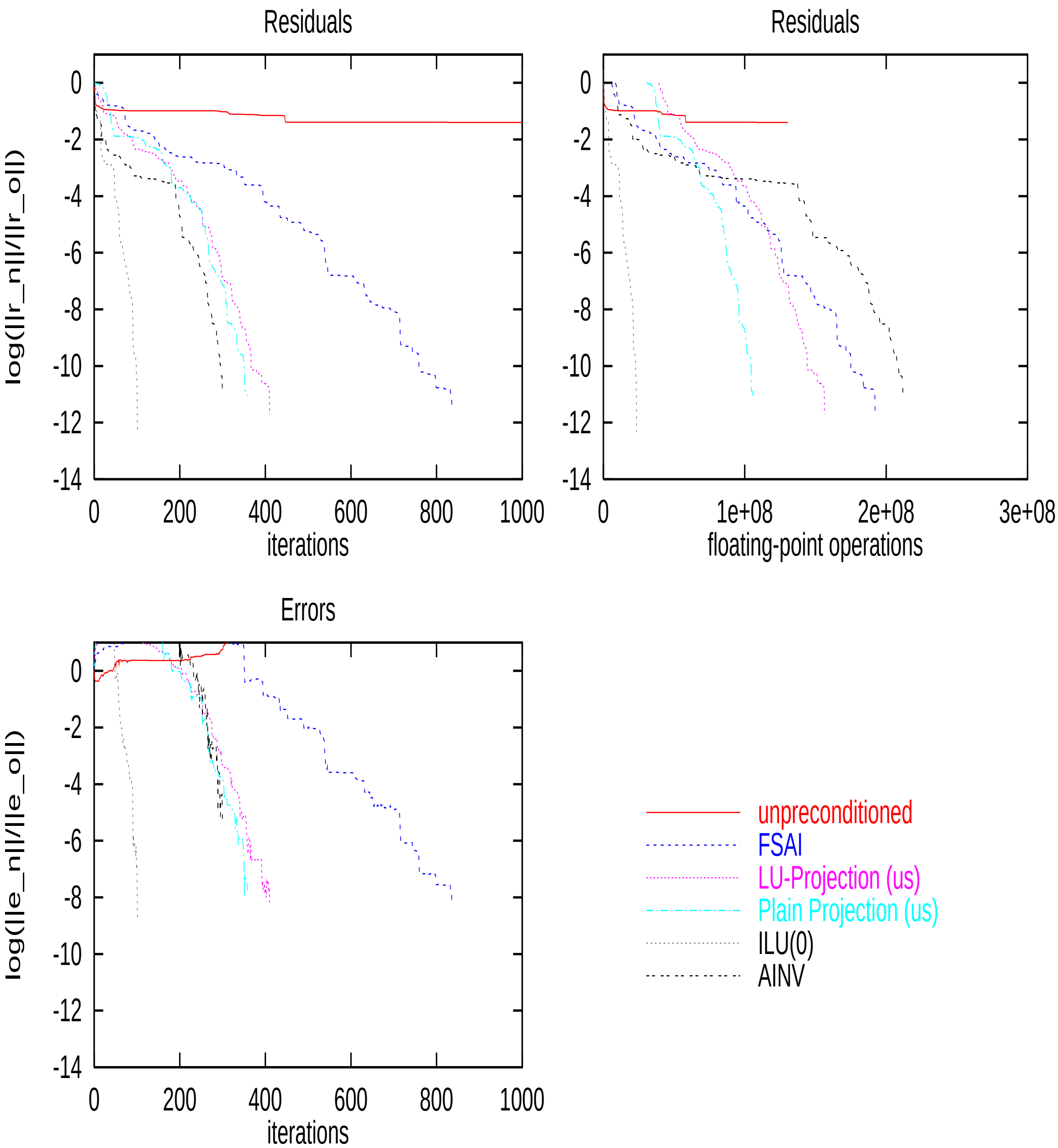


Figure 44: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix `utm1700b`.

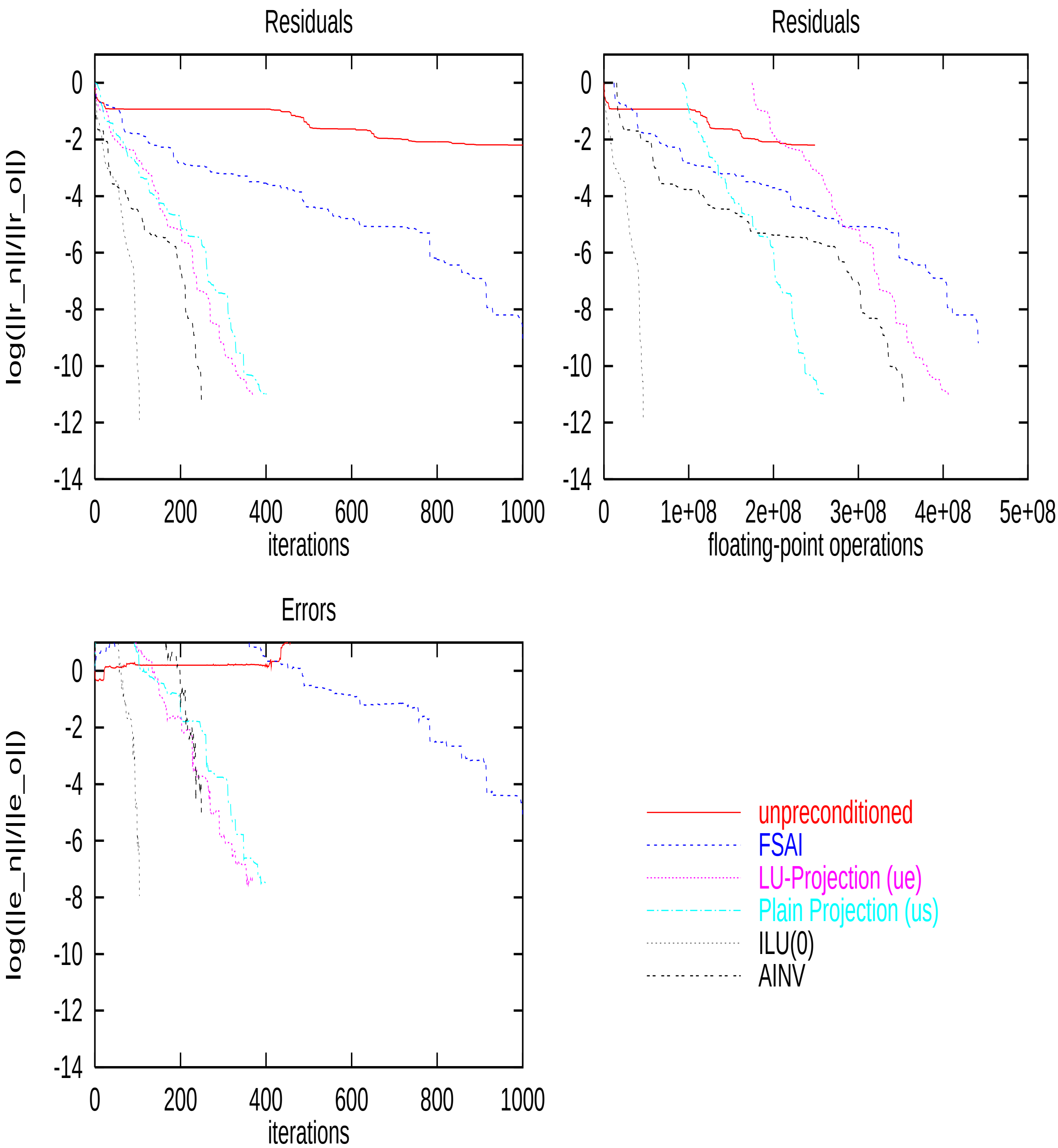


Figure 45: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix `utm3060`.

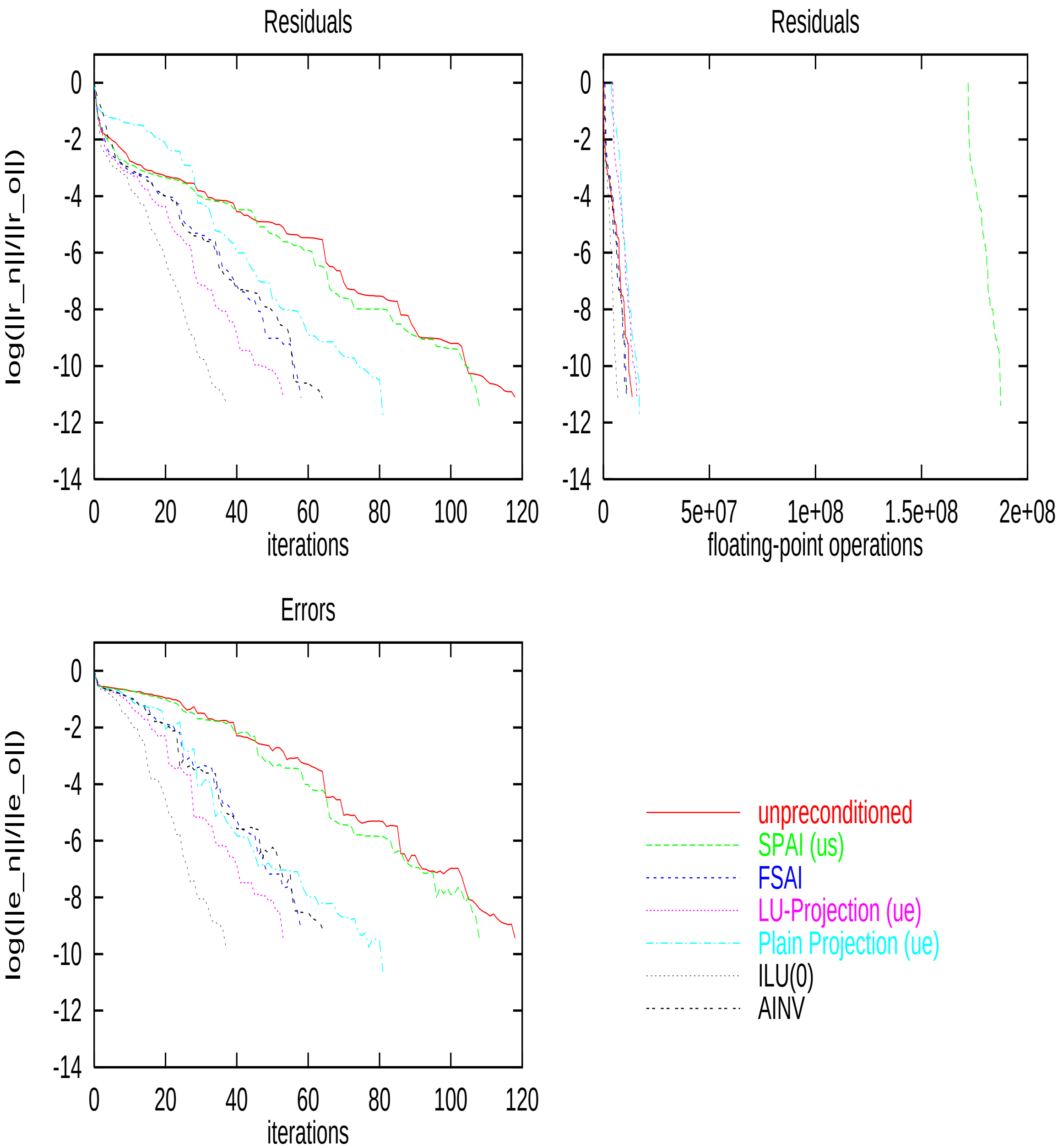


Figure 46: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix l_50_1.

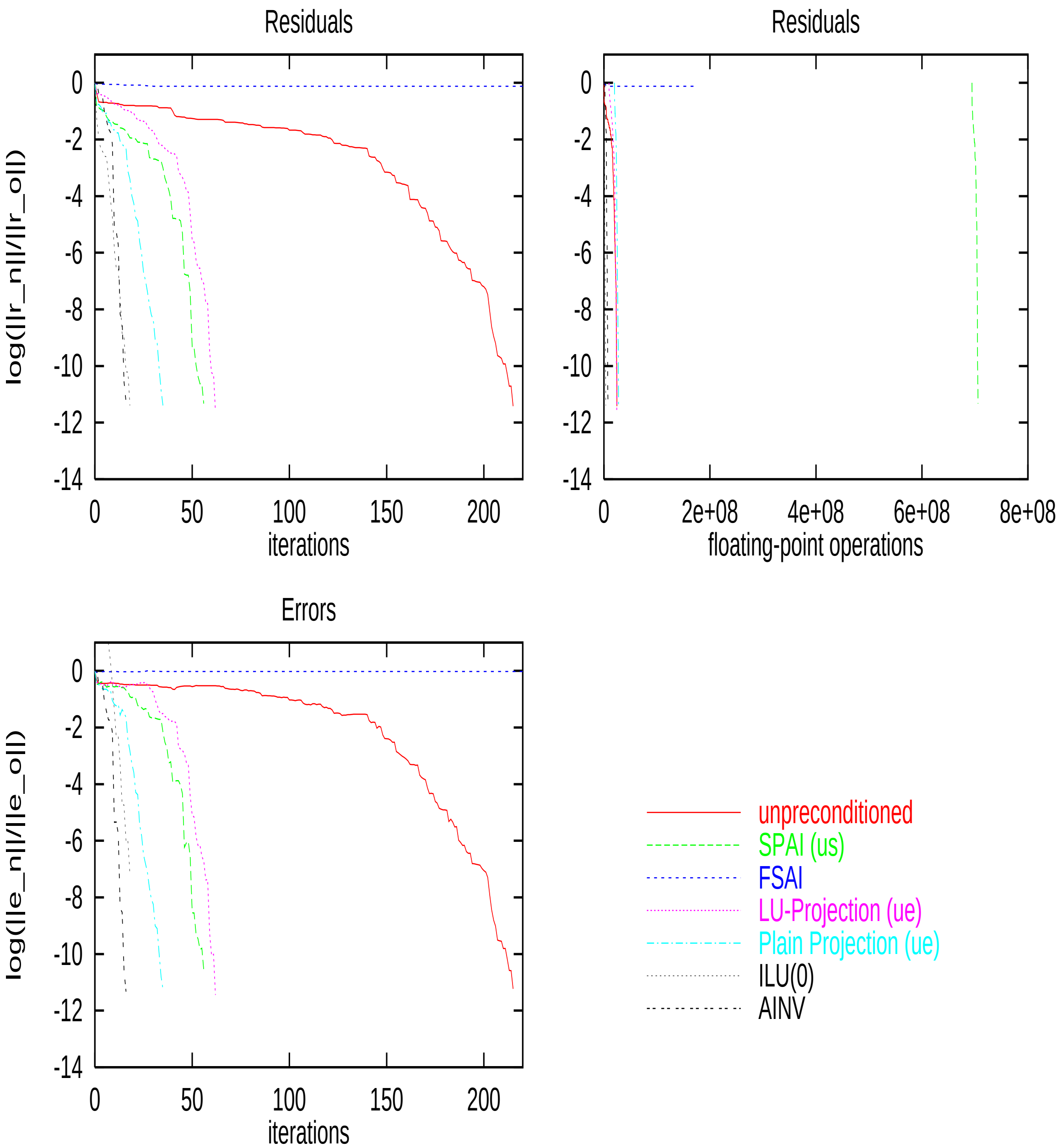


Figure 47: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix l_50_100.

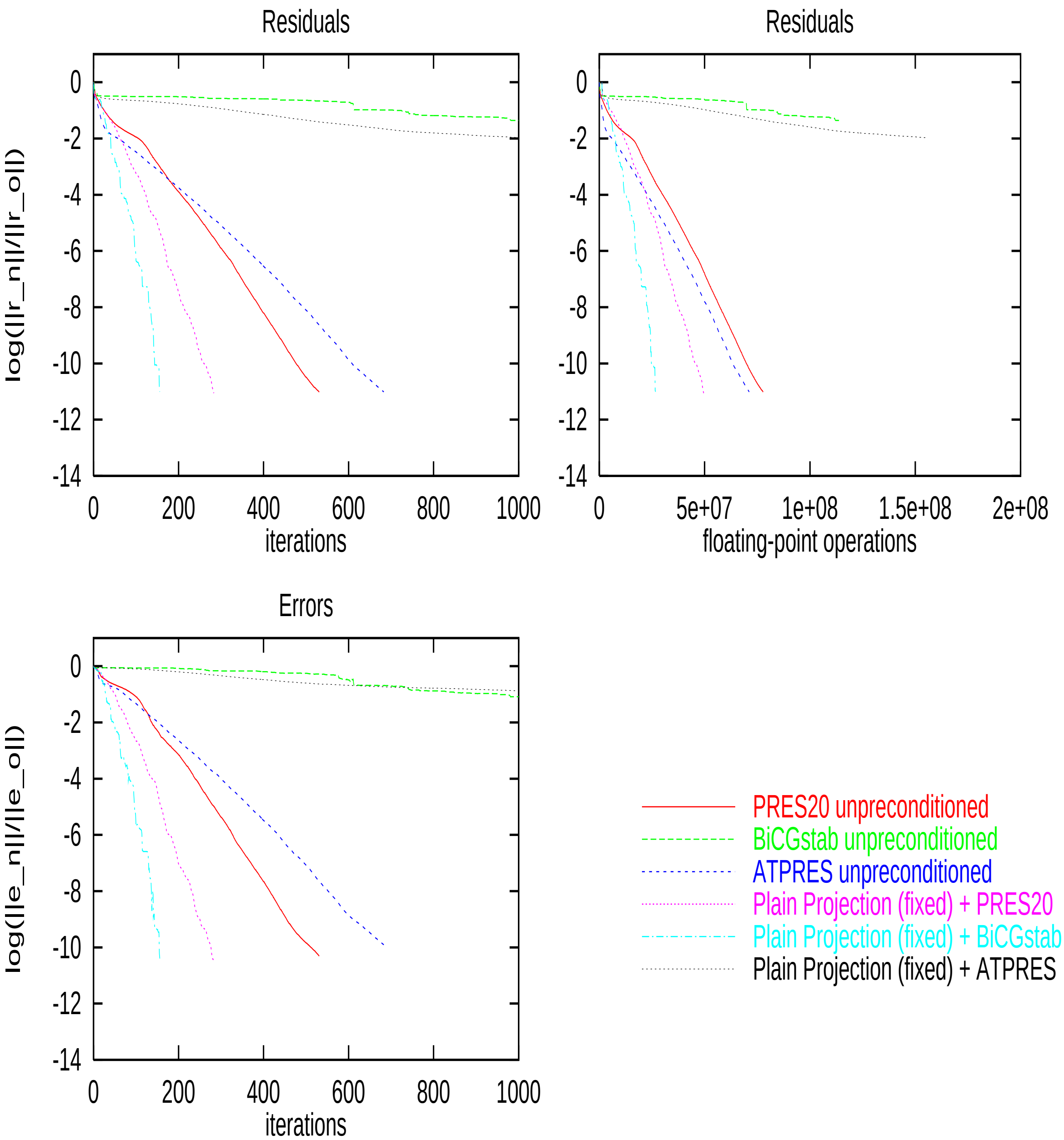


Figure 48: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix l_50_1000.

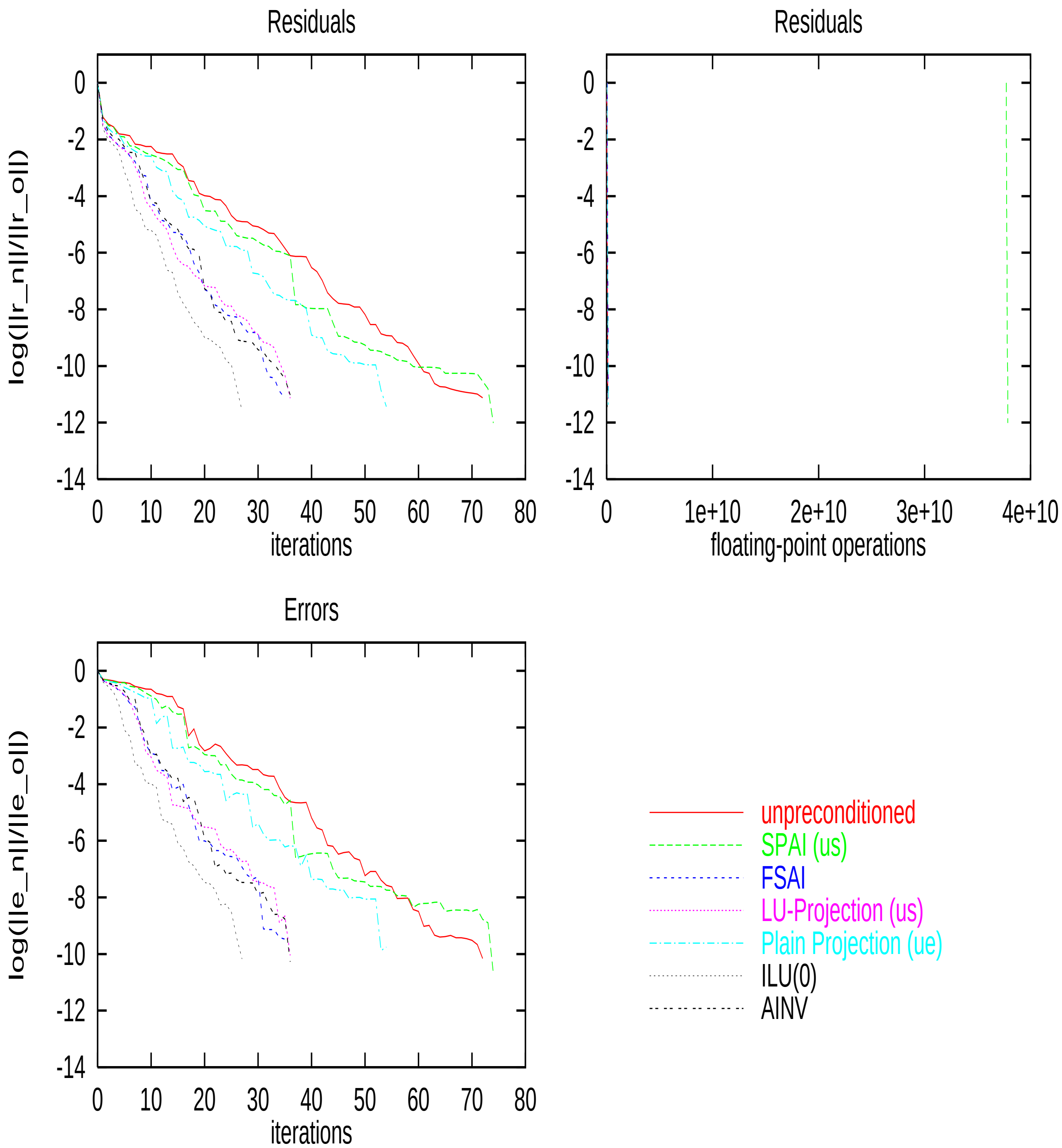


Figure 49: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix `c1m1o4`.

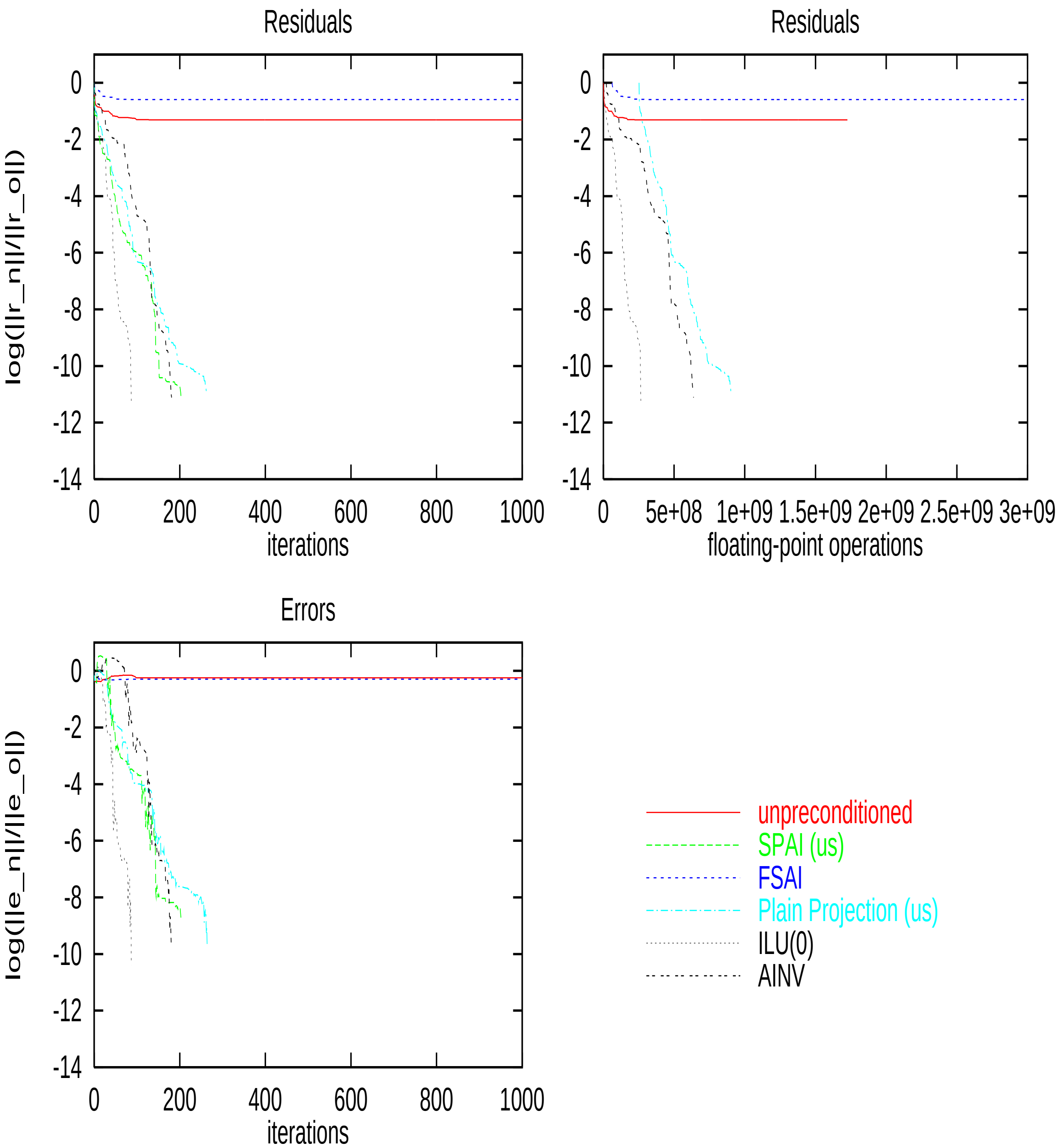


Figure 50: Comparison of the preconditioning methods for non-symmetric linear systems with the `BiCGstab` solver applied the matrix `c500m1o4`.

# 8 Summary and Outlook

In this chapter, a summary of the theoretical and practical results of this thesis is given. Further, open questions and problems are named.

In chapter 1, we have explained the general set of problems regarding the efficiency and robustness of iterative solvers, as well as the aim and the difficulties of promoting the robustness and efficiency of iterative solvers by preconditioning. The large number of problems and open questions in this area of today's scientific computing research was the motivation for this work.
In this thesis, we have considered projection methods – an entire class of inherently parallel methods for calculating approximate inverses – and we have considered the properties of particular projection methods applied as preconditioners for iterative solvers.
We have developed a theoretical framework for projection methods, including criteria and formulae for the explicit calculation of approximate inverses by projection methods. Importantly, we have deduced new approximation statements for general projection methods, which state on the distance between an approximate inverse calculated by a projection method and the true inverse. Based on these approximation statements, we have developed strategies for the adaptive derivation of sparsity patterns for approximate inverses calculated by projection methods. We have utilized this new theoretical framework for general projection methods in three ways:

i) We have classified some known preconditioning techniques, namely normalization (see section 6.1), `SPAI` (see section 6.2), `FSAI` (see section 6.7) and a further preconditioning technique proposed by Kolotilina and Yeremin in [46] (see section 6.3) in terms of projection methods and thereby we have provided the theoretical framework of projection methods for these preconditioning techniques.

ii) We have proposed strategies and algorithms for the adaptive derivation of sparsity patterns for general projection methods (algorithms 8 and 9, see chapter 5).

iii) We have obtained three new, inherently parallel preconditioning algorithms, namely `Plain projection`, $\mathtt{L^TL}$`-projection` and `LU-projection`, which are particular projection methods being capable of adaptively determining the sparsity pattern of the calculated approximate inverse.

In chapter 7, we compared the numerical properties of state-of-the-art preconditioning techniques (`ILU`, `IC`, `SPAI`, `AINV` and `FSAI`) to the properties of the three

newly proposed preconditioning techniques (`Plain projection`, $L^T$`L-projection`, `LU-projection`), which are projection methods with adaptive pattern derivation.
For the symmetric linear systems in our set of test problems, we compared `IC(0)`, `AINV`, `FSAI` and $L^T$`L-projection`. Basically, `FSAI` and $L^T$`L-projection` are obtained by the same projection method. But, while the `FSAI` algorithm utilizes the upper triangle of the original matrix as a fixed a priori pattern for the calculated approximate inverse, the $L^T$`L-projection` algorithm adaptively determines the pattern for the calculated approximate inverse. The results of our numerical tests indicate that the $L^T$`L-projection` algorithm is clearly superior to the `FSAI` method. Furthermore, in overall performance, the new $L^T$`L-projection` algorithm is comparable to `IC(0)` and `AINV`. Since $L^T$`L-projection` is inherently parallel, it is a promising new preconditioning technique.
For the unsymmetric linear systems, we compared `ILU(0)`, `SPAI`, `AINV`, `FSAI` to the newly proposed algorithms `Plain projection` and `LU-projection`. The results of these tests indicate that, as for robustness, `Plain projection` and `LU-projection` are comparable to the known methods `ILU(0)`, `SPAI` and `AINV`. Among all tested methods, the `SPAI` algorithm required by far the most floating-point operations for determining the approximate inverse. Therefore, and because the `Plain projection` and `LU-projection` algorithms offer even more potential for parallelization, these new methods are superior to `SPAI`.
Altogether, the numerical results discussed in chapter 7 indicate that the new pattern adaptive projection methods $L^T$`L-projection`, `Plain projection` and `LU-projection` are promising new preconditioning techniques, competitive to some of the best known preconditioning methods. In particular, because of the inherent parallelism of $L^T$`L-projection`, `Plain projection` and `LU-projection`, these new methods are well suited for parallel computers. However, exploring the properties of parallel implementations of $L^T$`L-projection`, `Plain projection` and `LU-projection` is beyond the scope of this thesis and will be subject to future research.
Further, the theoretical background for $L^T$`L-projection`, `Plain projection` and `LU-projection` can be further investigated. Conditions for the non-singularity of projective approximate inverses, as well as statements on the relation between the approximation statements given in chapter 4 and the convergence of the preconditioned iteration will be subject to supplemental research.
Since the algorithms based on $L^T$`L-projection`, `Plain projection` and `LU-projection` depend on several parameters, these algorithms are not easy to use. Further, if a suitable configuration of these parameters is found, it is an open question, whether these are optimal or not. Additionally, the relation between these parameters is unknown in general. Supplemental numerical testing accompanied by theoretical investigations on these problems is necessary.
The implementation of $L^T$`L-projection`, `Plain projection` and `LU-projection` allows numerous variants, e.g. the choice of the solver for the inner linear systems

and various strategies for numerical dropping in the pattern derivation process. Further tests on the influence of these choices on the performance of the preconditioned iterations are necessary.

Since preconditioning is particularly needed for ill-conditioned linear systems, and because determining sparse approximate inverses is an ill-posed problem, round-off errors caused by the finite precision of computers have to be taken into account. In this context, considering high precision computing and stable implementations of iterative solvers become topics. A sophisticated survey on high precision computing is given by Kaucher, Kulisch and Ullrich in [44]. Ample discussions on the stability of particular iterative solvers are given by Rozložník in [57], by Rozložník and Weiss in [58], by Drkošová, Rozložník, Strakoš and Greenbaum in [21], and by Greenbaum in [33].

# List of Tables



# List of Algorithms

# List of Figures

# References


[1] O. Axelsson and G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numer. Math.*, 48:499–523, 1986.

[2] M. W. Benson. Iterative solution of large scale linear systems. Thesis, Lakehead University, Thunder Bay, Canada, 1973.

[3] M. Benzi, J. K. Cullum, and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. Technical Report LA-UR-99-2899, Los Alamos National Laboratory, June 1999.

[4] M. Benzi, R. Kouhia, and M. Tůma. An assessment of some preconditioning techniques in shell problems. Technical Report LA-UR-97-3892, Los Alamos National Laboratory, September 1997.

[5] M. Benzi, J. Marín, and M. Tůma. A two-level parallel preconditioner based on sparse approximate inverses. In David R. Kincaid et. al., editor, *Iterative Methods in Scientific Computation II*, pages 1–11. IMACS, 1999.

[6] M. Benzi, C.D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17(5):1135–1149, 1996.

[7] M. Benzi, D. B. Szyld, and A. van Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems. Report 97-91, Department of Mathematics, Temple University, Philadelphia, PA 19122-2585, USA, 1997.

[8] M. Benzi and M. Tůma. A comparison of some preconditioning techniques for general sparse matrices. Dipartimento di Matematica, Università degli Studi di Bologna, 40127 Bologna, Italy.

[9] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. Dipartimento di Matematica, Università degli Studi di Bologna, 40127 Bologna, Italy.

[10] M. Benzi and M. Tůma. Orderings for factorized sparse approximate inverse preconditioners. Technical Report LA-UR-98-2175, Los Alamos National Laboratory, May 1998.

[11] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2–3):305–340, 1999.

[12] C. Brezinski. *Projection Methods for Systems of Equations.* Studies in Computational Mathematics. North Holland, Amsterdam, 1997.

[13] L. Brieger and G. Lecca. Parallel multigrid preconditioning of the conjugate gradient method for systems of subsurface hydrology. *J. Comput. Phys.*, 142(1):148–162, 1998.

[14] Z. Cai, T. A. Manteuffel, and S. F. McCormick. First-order system least squares for velocity-vorticity-pressure form of the Stokes equations, with application to linear elasticity. *Electronic Transactions on Numerical Analysis (ETNA)*, 3:150–159, December 1995.

[15] T. F. Chan, W. P. Tang, and W. L. Wan. Wavelet sparse approximate inverse preconditioners. Report CAM 97-34, Mathematics Dept., UCLA, 1997.

[16] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices. Research Report UMSI 94/101, University of Minnesota, Supercomputer Institute, 1200 Washington Avenue South, Minneapolis, Minnesota 55415, USA, 1994.

[17] J. D. F. Cosgrove, J. C. Diaz, and A. Griewank. Approximate inverse preconditionings for sparse linear systems. *Intern. J. Computer Math.*, 44:91–110, 1992.

[18] J. W. Daniel, W. B. Gragg, L. Kaufmann, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Mathematics of Computation*, 30(136):772–795, 1976.

[19] T. Davis. University of florida sparse matrix collection, http://www.cise.ufl.edu/~davis/sparse/, ftp://ftp.cise.ufl.edu/pub/faculty/davis/matrices/. *NA Digest*, Volume 97(no. 23), June 7, 1997.

[20] V. R. Deshpande, M. J. Grote, P. Messmer, and W. B. Sawyer. Parallel sparse approximate inverse preconditioner. Technical Report TR-96-14, Swiss Center for Scientific Computing (CSCS/SCSC), via Cantonale, 6928 Manno, Switzerland, 1996.

[21] J. Drkošová, M. Rozložník, Z. Strakoš, and A. Greenbaum. Numerical stability of GMRES. *BIT*, 3:309–330, 1995.

[22] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14, 1989.

[23] I.S. Duff. Permutations for a zero-free diagonal. *ACM Transactions on Mathematical Software*, 7(3):387–390, September 1981.

[24] M. R. Field. Improving the performance of factorised sparse approximate inverse preconditioner. Technical Report HDL-TR-98-199, Hitachi Dublin Laboratory, 1998.

[25] M. R. Field. A parallel factorised sparse approximate inverse preconditioner with improved choice of sparsity pattern. Technical Report HDL-TR-99-214, Hitachi Dublin Laboratory, 1999.

[26] R. W. Freund. On conjugate gradient type methods and polynomial preconditioners for a class of non-Hermitian matrices. *Numer. Math.*, 57:285–312, 1990.

[27] S. Fujino and S. Doi. Optimizing multicolor ICCG methods on some vectorcomputers. In R. Beauwens and P. de Groen, editors, *Iterative Methods in linear Algebra*, pages 349–358. Elsevier Science Publishers B. V. (North-Holland), 1992.

[28] S. Fujino and T. Takeuchi. ILU factorization well suited to the vector processor using a variant of the 5-point difference scheme. *Comp. Phys. Comm.*, 85(3):371–381, 1995.

[29] S. Fujino, S. Zhang, and M. Mori. Visualization of convergence behavior of Bi-CGSTAB method. *Supercomputer*, 8(6):127–135, 1992.

[30] R. Glowinski, A. Rieder, R. Wells jr., and X. Zhou. A wavelet multigrid preconditioner for dirichlet boundary value problems in general domains. *Modelisation Math. Anal. Numer.*, 30(6):711–729, 1996.

[31] G. Golubovici and C. Popa. Interpolation and related coarsening techniques for the algebraic multigrid methods. In *Proceedings of the Fourth European Conference on Multigrid Methods, July 1993, Amsterdam*, volume 116 of *Intern. Series of Numer. Math*, pages 201–213, Berlin, 1994. Birkhäuser Verlag.

[32] N.I.M. Gould and J. Scott. Sparse approximate-inverse preconditioners using norm-minimization techniques. *SIAM J. Sci. Comput.*, 19(2):605–625, 1998.

[33] A. Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, 1997.

[34] I. Gustafsson. A class of first order factorization methods. *BIT*, 18(1):142–156, 1978.

[35] W. Hackbusch. *Multigrid Methods and Applications.* Springer-Verlag, Berlin, 1985.

[36] D. R. Hare, C. R. Johnson, D. D. Olesky, and P. van den Driessche. Sparsity analysis of the QR factorization. *SIAM J. Matrix Anal. Appl.*, 14(3):655–669, 1993.

[37] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–435, 1952.

[38] T. Huckle. Approximate sparsity patterns for the inverse of a matrix and preconditioning. *Applied Numerical Mathematics*, 30(2–3):291–303, 1999.

[39] T. Huckle and M. Grote. A new approach to parallel preconditioning with sparse approximate inverses. Internal Report SCCM94-03, Stanford University, 1994.

[40] V. P. Il'in. *Iterative Incomplete Factorization Methods*. World Scientific Publishing Co. Pte. Ltd, Singapore, 1992.

[41] O. G. Johnson, C. A. Micchelli, and G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM J. Numer. Anal.*, 20(2):362–376, 1983.

[42] W. D. Joubert and D. Manteuffel. Iterative methods for nonsymmetric linear systems. In D. R. Kincaid and L. J. Hayes, editors, *Iterative Methods for Large Linear Systems*, pages 149–171. Academic Press, Boston, 1990.

[43] I.E. Kaporin. High quality preconditioning of a general symmetric positive definite matrix based on its $u^t u + u^t r + r^t u$-decomposition. *Numer. Linear Algebra Appl.*, 5:483–509, 1998.

[44] E. Kaucher, U. Kulisch, and C. Ullrich. *Computerarithmetic*. Teubner-Verlag, Stuttgart, 1987.

[45] S.A. Kharchenko, L.Yu. Kolotilina, A.A. Nikishin, and A.Yu. Yeremin. A robust AINV-type preconditioning method for constructing sparse approximate inverse preconditioners in factored form. To appear in Numerical Linear Algebra and Applications.

[46] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings i. theory. *SIAM J. Matrix Anal. Appl.*, 14(1):45–58, 1993.

[47] L. A. Krukier. Convergence of triangular iterative methods based on the skew-symmetric part of the matrix. *Applied Numerical Mathematics*, 30(2–3):281–290, 1999.

[48] G. I. Marchuk. *Methods for Calculating the Nuclear Reactors*. Atomizdat, Moskva, 1958. In Russian language.

[49] U. Meier Yang. Preconditioned conjugate gradient-like methods for nonsymmetric linear systems. Tech. Report CSRD 1210, University of Illinois at Urbana-Champaign, Center for Supercomputing Research and Development, 465 CSRL-1308 West Main Street, Urbana, IL 61801-2307, USA, 1992.

[50] U. Meier Yang and K. A. Gallivan. A new family of preconditioned iterative solvers for nonsymmetric linear systems. *Appl. Numer. Math.*, 19(3):287–317, 1995.

[51] U. Meier Yang and K.A. Gallivan. A new family of block methods. *Applied Numerical Mathematics*, 30(2–3):155–173, 1999.

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

[53] N. S. Mendelsohn. Some elementary properties of ill conditioned matrices and linear equations. *American Mathematical Monthly*, 63:285–295, 1956.

[54] Gérard Meurant. *Computer Solution of Large Linear Systems.* Number 28 in Studies in Mathematics and its Applications. North Holland, Amsterdam, 1999.

[55] C. Popa. Preconditioning for the fulfilment of the approximation assumption in the algebraic multigrid method. *Stud. Univ. Babes-Bolyai*, 40(1):77–102, 1995.

[56] L. Portugal, F. Bastos, J. Judice, J. Paixão, and T. Terlaky. An investigation of interior point algorithms for the linear tranportation problem. Report 93-100, Delft University of Technology, Faculty of Technical Mathematics and Informatics, 1993.

[57] M. Rozložník. *Numerical Stability of the GMRES Method.* PhD thesis, Academy of Sciences of the Czech Republik, Institute of Computer Science, Prague, 1997.

[58] M. Rozložník and R. Weiss. On stable implementations of the generalized minimal error method. *J. Comp. and Appl. Math.*, 98(1):49–62, 1998.

[59] Y. Saad. Preconditioning techniques for nonsymmetric and indefinite linear systems. *J. Comput. Appl. Math.*, 24:89–105, 1988.

[60] Y. Saad. ILUT: A dual threshold incomplete LU factorization. Research Report UMSI 92/38, University of Minnesota, Supercomputer Institute, 1200 Washington Avenue South, Minneapolis, Minnesota 55415, USA, 1992.

[61] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

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

[63] J. Schoeberl. Robust multigrid preconditioning for parameter-dependent problems. i: The stokes-type case. In W. Hackbusch (ed.) et al., editor, *Multigrid methods V. Proceedings of the 5th European multigrid conference, held in Stuttgart, Germany, October 1–4, 1996*, pages 260–275. Berlin: Springer. Lect. Notes Comput. Sci., 1998.

[64] W. Schönauer. *Scientific Computing on Vector Computers.* North-Holland, Amsterdam, New York, Oxford, Tokyo, 1987.

[65] W. Schönauer and H. Häfner. Supercomputers: The hardware, the architecture. In R. Vichnevetsky and J. J. H. Miller, editors, *Proceedings of the 13th IMACS World Congress on Computation and Applied Mathematics, Dublin, Ireland, July 22-26*, pages 725–727, 1991.

[66] W. Schönauer, H. Müller, and E. Schnepf. Pseudo-residual type methods for the iterative solution of large linear systems on vector computers. In M. Feilmeier, J. Joubert, and U. Schendel, editors, *Parallel Computing 85*, pages 193–198. Elsevier Science Publishers B. V. (North-Holland), 1986.

[67] W. Schönauer and K. Raith. A polyalgorithm with diagonal storing for the solution of very large indefinite linear banded systems on vector computers. In M. Ruschitzka, editor, *Parallel and Large-Scale Computers: Performance, Architecture, Applications*, volume II of *IMACS Transaction on Scientific Computation*, pages 213–220. North-Holland, 1983.

[68] W. Schönauer, M. Schlichte, and R. Weiss. Wrong ways and promising ways towards robust and efficient iterative linear solvers. In *Advances in Computer Methods for Partial Differential Equations VI*, pages 7–14. Publ. IMACS, 1987.

[69] W. Schönauer, E. Schnepf, and H. Müller. The FIDISOL program package. Internal report 27/85, University of Karlsruhe, Computing Center, Postfach 6980, 76128 Karlsruhe, Germany, 1985.

[70] J. R. Shewchuk. Conjugate gradient method without the agonizing pain. Internal Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 1994.

[71] G. L. G. Sleijpen and D. R. Fokkema. BiCGStab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electronic Transactions on Numer. Anal. (ETNA)*, 1:11–32, 1993.

[72] G. L. G. Sleijpen and H. A. van der Vorst. Maintaining convergence properties of BiCGstab methods in finite precision arithmetic. *Numerical Algorithms*, 10(3–4):203–223, 1995.

[73] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 10(1):36–52, 1989.

[74] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, Berlin, second edition, 1993.

[75] F. Stummel and K. Hainer. *Praktische Mathematik.* Teubner-Verlag, Stuttgart, 1982.

[76] H. A. van der Vorst. Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *J. Comput. Phys.*, 44:134–155, 1981.

[77] H. A. van der Vorst. *Preconditioning by Incomplete Decompositions.* PhD thesis, University of Utrecht, The Netherlands, 1982. ACCU Series 32.

[78] H. A. van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.

[79] R. Weiss. Error-minimizing Krylov subspace methods. *SIAM J. Sci. Comput.*, 15(3):511–527, 1994.

[80] R. Weiss. *Parameter-Free Iterative Linear Solvers.* Mathematical Research, vol. 97. Akademie Verlag, Berlin, 1996.

[81] R. Weiss, H. Häfner, and W. Schönauer. LINSOL (LINear SOLver)—description and user's guide for the parallelized version. Internal report 61/95, University of Karlsruhe, Computing Center, Postfach 6980, 76128 Karlsruhe, Germany, 1995.

[82] G. Wittum and F. Liebau. On truncated incomplete decompositions. *BIT*, 29:719–740, 1989.

[83] M. Zimmermann. Projektionsverfahren zur Präkonditionierung von verallgemeinerten CG-Verfahren. Internal report 55/95, University of Karlsruhe, Computing Center, Postfach 6980, 76128 Karlsruhe, Germany, 1995. Diploma thesis.

# Lebenslauf

| | |
|---|---|
| Name: | Claus Peter Koschnski |
| Gebohren: | am 18. März 1970 in Essen |
| Familienstand: | ledig |
| 1976 – 1980 | Dionysius Grundschule in Essen Borbeck |
| 1980 – 1989 | Don-Bosco Gymnasium in Essen Borbeck |
| 1989 | Abitur |
| 1989 – 1990 | Grundwehrdienst |
| 1990 – 1995 | Studium der Mathematik an der Universität (GHS) Essen |
| Oktober 1995 | Diplomprüfung |
| Nov. 1995 – März 1996 | wissenschaftlicher Angestellter am Institut für experimentelle Mathematik der Universität (GHS) Essen |
| April 1996 – Juli 1998 | Mitarbeiter in der Abteilung Netzwerke des Rechenzentrums der Universität (TH) Karlsruhe |
| April 1996 – Oktober 1999 | Anfertigung der Dissertatrion *Properties of Approximate Inverses and Adaptive Control Concepts for Preconditioning* in der Forschungsgruppe "Numerikforschung für Supercomputer" am Rechenzentrum der Universität (TH) Karlsruhe |