Faculty of Mechatronics,
Informatics and Interdisciplinary Studies

Technical University of Liberec, Liberec

Institute of Computer Science

Academy of Sciences of the Czech Republic, Prague

Generalized Gram–Schmidt Process: Its Analysis and Use in Preconditioning

Jiří Kopal

Report of the PhD Thesis, June 2014

Report of the PhD Thesis
Generalized Gram–Schmidt Process: Its Analysis and Use in Preconditioning

Author:	Jiří Kopal (`jiri.kopal@tul.cz`)
Supervisor:	Miroslav Tůma (`tuma@cs.cas.cz`)
Supervisor specialist:	Miroslav Rozložník (`miro@cs.cas.cz`)
Study programme:	P2612 Electrotechnics and informatics
Field of study:	3901V025 Science engineering
Departments:	Faculty of Mechatronics, Informatics and Interdisciplinary Studies Institute of Novel Technologies and Applied Informatics, Technical University of Liberec, Studentská 2, 461 17 Liberec 1, Czech Republic
and	Institute of Computer Science, Department of Computational Methods, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic

Abstract

Many problems arising from mathematical modelling in science and technology lead to solving large and sparse systems of linear algebraic equations. Then a natural way to solve them is to use iterative methods based on Krylov subspaces such as the method of conjugate gradients. In order to be efficient, such methods often need good preconditioners.

We focus on a particular strategy to obtain such preconditioners based on the generalized Gram–Schmidt process. We study both its theoretical properties as well as algorithms to compute it approximately and use the resulting factors as preconditioners. In this way, the thesis couples together two different areas of numerical linear algebra, i.e., numerical analysis and computational mathematics that is more focused on real-world computations.

From the theoretical point of view, the generalized Gram–Schmidt process in exact arithmetic computes a factorization of the inverse of the corresponding matrix. In other words, it can be formally considered as a direct method. Here we prefer to see the process as an incomplete scheme that produces a factorized sparse approximate inverse of the matrix. The generalized Gram–Schmidt process was introduced in a couple of important classical papers as [16, 23]. Its use as an incomplete scheme for computing approximate inverse preconditioning was proposed by Benzi et al. [2] and later enhanced in [1] (2000). The incompleteness is achieved by a dropping strategy based on discarding entries that are in some sense small.

We analyze the generalized Gram–Schmidt process in finite precision arithmetic. This analysis is a continuation of the paper by Rozložník et al. [31] (2012). For example, we extend the generalized Gram–Schmidt process by pivoting that enables an improvement of some error bounds. In particular, differences between component-wise and norm-wise error bounds are highlighted.

Both theoretical considerations as well as experimental observations lead to a new dropping technique that behaves similarly to rounding in finite precision arithmetic. The new dropping technique introduced by Kopal et al. [26] (2013) is studied more in detail and derived in a different way.

The main goal is to present a new dropping technique that may generally help to better understand of the interplay between floating-point analysis and numerical aspects of preconditioning by incomplete decompositions.

Keywords: approximate inverse preconditioning, Gram–Schmidt process, pivoting, sparse matrices, incomplete algorithms, dropping techniques, iterative methods

Abstrakt

Mnoho úloh matematického modelování ve vědě a technice vede na řešení velkých a řídkých soustav lineárních algebraických rovnic. Takové úlohy je přirozené řešit pomocí iteračních metod založených na Krylovovských podprostorech, což je například metoda sdružených gradientů. Aby dané řešení bylo efektivní, iterační metody potřebují dobré předpodmínění.

Zaměříme se na na konkrétní strategii výpočtu takových předpodmínění, které jsou založeny na zobecněném Gram–Schmidtově procesu. Zabýváme se jak studiem teoretických vlastností úplného algoritmu tak i jeho neúplnou verzí, kterou používáme pro výpočet faktorů předpodmínění. Tímto způsobem se zde prolínají dvě různé disciplíny numerické lineární algebry, numerickou analýzu a výpočetní matematiku se zaměření na reálné výpočty.

Z teoretického hlediska Gram–Schmidtův algoritmus v přesné aritmetice počítá faktorizaci inverze příslušné matice. Může být proto formálně považován za přímou metodu. Zde se budeme zabývat především jeho neúplnou verzí, coby algoritmem pro výpočet přibližné řídké faktorizace inverze matice. Zobecněný Gram–Schmidtův proces byl představen v článcích [16, 23]. Jako neúplný algoritmus pro výpočet přibližné inverze předpodmínění byl navržen Benzim et al. [2] a později byl koncept rozšířen v [1]. Neúplnost je realizována technikou odvrhování prvků, ktere jsou v nějakém smyslu malé.

Zabýváme se analýzou zobecněného Gram–Schmidtova procesu v aritmetice s konečnou přesností. Analýza navazuje na článek Rozložníka et al. [31] (2012), zde je rozšířena například o pivotaci, která umožnila vylepšit některé horní odhady chyb. Zejména rozdíly mezi horními odhady chyb po prvcích a v normě jsou zdůrazňovány.

Jak teoretické tak experimentální úvahy daly základ nové technice odvrhování prvků, která vykazuje stejné vlastnosti jako zaokrouhlování v aritmetice s konečnou přesností. Tato nová technika odvrhování představená Kopalem et al. [26] (2013) je zde studována detailněji a odvozena jiným způsobem.

Hlavním cílem je prezentovat techniku odvrhování prvků, která může obecně napomoci lepšímu pochopení spojitosti mezi analýzou v aritmetice s konečnou přesností a numerickými aspekty predpodmínění počítaných pomocí neúplných rozkladů.

Klíčová slova: předpodmínění přibližnými inverzemi, Gram–Schmidtův proces, pivotace, řídké matice, neúplné algoritmy, odvrhovací techniky, iterační metody

Contents

Chapter 1

Introduction

1.1 A brief background

A lot of challenging problems in science and industrial applications lead to large systems of linear algebraic equations. It happens very often that these systems are sparse. Such applications include, for example, modeling in climate forecasts, geology, quantum chemistry, circuit design, aerospace computations and many others. One needs to get a solution of acceptable accuracy within a reasonably short time. This is especially true in real-time simulations. An example is the problem of weather forecast where the need to obtain results very fast is absolutely crucial. Computation of large and sparse systems of linear algebraic equations that arise from discretization partial differential equation (PDE) is another important area with the need to solve systems very quickly in particular if they result from sequences from solving nonlinear problems.

There are two basic classes of methods for solving large and sparse systems of linear algebraic equations: direct and iterative ones. Direct methods have been developed over time from the basic scheme of the Gaussian elimination. Their main idea is to solve the system as accurately as the computational hardware and software allow in a finite number of steps. This often requires a very large amount of work. Iterative methods represent a wide class of methods that obtain the solution by generating a sequence of successive approximations. They are appropriate when the direct methods are prohibitively expensive and, sometimes, they are the only possible choice in practice. An important subclass of iterative methods is represented by the Krylov subspace methods. This subclass is based on forming a basis of the subspace that involves successive powers of the system matrix and gets an approximate solution typically from solving a certain low-dimensional optimization problem. The complicated nonlinear nature of the process implies that their convergence behavior may be not fully understood. In general, one can say that direct methods come with robustness but at the expense of additional computational work. On the other hand, iterative methods provide very often a sequence of cheap steps but they may suffer from slow convergence or even stagnation.

Numerical methods are also strongly linked with computer architectures. Although different computers perform various operations of linear algebra with different efficiencies, a common feature shared by the vast majority of computers is that quantities are computed only approximatively with a finite precision. The knowledge of the arithmetic is often important in design and application of numerical methods.

1.2 Scientific framework of the thesis

In many applications there is no need to find highly accurate solutions of their subproblems. Instead, obtaining their rough approximations is sufficient. This is an important reason why iterative methods may be methods of choice. However, in order to increase their robustness, the problems should be suitably transformed. Such a transformation is called preconditioning and it should lead to solving problems that are more tractable by the iterative method than the original ones. Note that preconditioning may mean completely different transformations for different problems with their specific properties.

Properties of systems of linear algebraic equations depend on several factors, e.g., on physics of the considered application or methods of the problem discretization. In our description we concentrate on a compact introduction into preconditioning that covers just the case when the system matrix is symmetric and positive definite (SPD). Main contributions of this thesis restrict to this case.

We consider the generalized Gram–Schmidt process (GGS), that is the Gram–Schmidt process with energetic inner product induced by an SPD matrix A. Numerical behavior of all main orthogonalization schemes including Householder, Givens QR, and standard modified and classical Gram–Schmidt decompositions is well understood for $A = I$ [24, 9, 17, 18, 33]. This is not the case, however, for generalized schemes with $A \neq I$. For example, the QR decomposition for solving weighted least squares problems is studied in [21, 19, 20]. The modified QR decomposition with a non-standard inner product with an SPD matrix, where matrix is given in the factored form as $A = LL^T$ (its Cholesky factor is known a priori) was studied in [38], see also, [37, 36]. Several orthogonalization schemes are analyzed in [31] and also in [29].

Independently of the papers on the Gram–Schmidt process, an incomplete decomposition based on the general oblique GGS orthogonalization process has been introduced in [2]. It is called *AINV* (Approximate INVerse) and is based on a generalization of the orthogonalization scheme that is close to the classical Gram-Schmidt process. Further development of AINV has led to its stabilized variant so-called *SAINV* [1]. The stabilization was performed both in terms of the orthogonalization scheme (changed to the modified GGS process) and in terms of appropriate computation of the normalization coefficients (one sided (non-stabilized) versus stabilized [1, 4]). It was demonstrated that both AINV and SAINV may provide successful preconditioning, in particular for some classes of problems. On the other hand, process to obtain appropriate dropping parameters that balance sparsity and accuracy of the approximate inverse has been restricted to a *trial-and-error* approach.

Up to now, no sufficiently general theory for preconditioning by incomplete factorizations has been proposed. Research in this field even for preconditioning of general SPD problems is focused either to analyzing the behavior of algorithms in finite precision arithmetic or to development of incomplete schemes and it covers both theoretical and practical aspects of the problem very rarely. This note applies to approaches based on both the standard or generalized Gram–Schmidt process. It may be caused by the fact that an analysis of the most common dropping techniques that control discarding of the small entries is difficult. Typically, errors that arise from dropping modify the schemes in a different way than errors due to rounding in the floating-point arithmetic.

In contrast to the SPD case, there exist nice contributions devoted to its special cases (such as finite difference matrices). An important work devoted to the modified incomplete Cholesky factorization [22] that analyzes the condition number of the preconditioned system was presented by Ivar Gustafsson. An even earlier pioneering work is due to Dupont et al. [15]. Another theoretical approach deals with the support graph preconditioning [10, 5].

Lack of theory for preconditioning by incomplete decompositions has been our main motivation. We apply the approach based on rounding error analysis to preconditioning, although it uses only one specific preconditioning strategy, namely that based on the generalized Gram–Schmidt process. The theory and experimental observations result into a new dropping strategy that seems to be rather efficient. A set of numerical experiments then demonstrates that additional techniques as scaling and pivoting may significantly improve properties of the preconditioner. Last but not least, we believe that analyzing floating-point properties of decompositions may also lead to development of other types of algebraic preconditioners.

1.3 Outline of the thesis

The thesis is divided into four parts. The first part is the introduction, it also involves chapter introduction, that brings an insight into main topic of the thesis. The second part is devoted to solve the systems of linear algebraic equations with a symmetric and positive definite matrix. We provide a survey of direct and iterative solution methods and preconditioning techniques. The third part is focused on the generalized Gram-Schmidt process. We focus to the exact arithmetic identities and pivoting, we also deal with analysis in finite precision arithmetic, we verify theoretical results on simple numerical experiments, we introduce a new approach of construction of the incomplete schemes and finally we present numerical experiments that verifies properties of the previously introduced incomplete scheme. The fourth part contains summary, possible directions of the research in the future, open questions and also the list of related publications of the author.

1.4 Finite precision arithmetic

Representation of the real numbers is on computer restricted to a finite subset. More details can be found in IEEE 754 standard (current version IEEE 754-2008) that defines the floating point arithmetic. Probably the mostly used data type for scientific application is data type with double precision that uses 64 bit data representation that corresponds to the relative machine precision $\epsilon \approx 1.1 \cdot 10^{-16}$ (and the unit roundoff $u = 2\epsilon \approx 2.2 \cdot 10^{-16}$). IEEE 754 standard also guaranties result of the each elementary operation $(+, -, \times, /\ )$ and also for the square root $\sqrt{\cdot}$ is correctly rounded value of the exact operation.

Consider two real numbers α and β and an elementary floating point operation $\mathrm{fl}[\cdot \text{ op } \cdot]$ in finite precision arithmetic. If the operation is well defined it holds

$$|\mathrm{fl}[\alpha \text{ op } \beta]| \leq (\alpha \text{ op } \beta)(1+\delta), \quad |\delta| \leq u, \tag{1.1}$$

similarly also for square root. The standard IEEE 754 also includes extensive recommendations for advanced exception handling, additional operations (such as trigonometric functions), expression evaluation, and for achieving reproducible results.

Using (1.1) one can establish error bounds for various operations of linear algebra. E.g., for matrix-vector multiplication Mv, where $M \in \mathbb{R}^{m\times n}$ and $v \in \mathbb{R}^n$, we have

$$|\mathrm{fl}[Mv] - Mv| \leq \mathcal{O}(n)u|M||v|, \tag{1.2}$$

where $|\cdot|$ denotes the absolute value. Note that this notation can be used not only for scalars, but also for vectors and matrices. The expression $\mathcal{O}(n)$ denotes low-degree polynomials in n. Rounding error bounds for the matrix-vector multiplication can be also formulated by using spectral and Euclidean norms of the matrices and vectors, respectively, (both denoted by $\|\cdot\|$) as

$$\|\mathrm{fl}[Mv] - Mv\| \leq \mathcal{O}(n^{3/2})u\|M\|\|v\|. \tag{1.3}$$

Rounding errors for additional linear algebra operations are developed in Chapter 3. In general, for upper bound for rounding errors, coefficients of the polynomial (e.g., in (1.2), (1.3)) in terms $\mathcal{O}(\cdot)$ are very small. Norm-wise error bounds as (1.3) may to overestimate actual result. Throughout the thesis, we use only the bounds for the rounding errors that are linear in the unit roundoff u and do not consider the higher order terms.

1.5 System of linear algebraic equations

A system of linear algebraic equations can be written in the form

$$\sum_{j=1}^{m} a_{i,j} x_j = b_i, \quad i \in 1, \dots, n$$

where the coefficients $a_{i,j}$ are scalars. The solution components are denoted by $x_1, x_2, \dots, x_n$ and $b_1, b_2, \dots, b_n$ denote the components of the right-hand side vector. The system can be written in a more compact form using the matrix notation as

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m, \tag{1.4}$$

$$A = [a_{i,j}] = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Here we will always deal with the case where A is a real, square ($m \equiv n$) and *nonsingular* matrix ($\mathcal{R}(A) = \mathbb{R}^n$). Let us characterize the three main classes of such matrices by their properties and properties of their singular values $\sigma_i(A)$ and eigenvalues $\lambda_i(A)$:

1. A is symmetric and positive definite (SPD)
   - $(\forall i, j)(a_{i,j} = a_{j,i})$ and
   - $(\forall i)(\lambda_i(A) > 0) \Leftrightarrow (\forall x \neq 0)(x^T A x > 0)$,
   - note that in this case we always have $(\forall i)(\sigma_i(A) = \lambda_i(A))$.
2. A is symmetric and generally indefinite
   - $(\forall i, j)(a_{i,j} = a_{j,i})$ thus
   - $(\forall i)(\lambda_i(A) \in \mathbb{R} \setminus \{0\})$,
   - similarly as above, we have $(\forall i)(\sigma_i(A) = |\lambda_i(A)|)$.
3. A is non-symmetric
   - $(\forall i, j)(a_{i,j} \lesseqqgtr a_{j,i})$,
   - $(\forall i)(\lambda_i(A) \in \mathbb{C} \setminus \{0 + 0\,\mathbf{i}\})$,
   - $(\forall i)(\sigma_i(A) \lesseqqgtr |\lambda_i(A)|)$.

We mostly consider solving systems of equations with symmetric and positive definite matrices.

Chapter 2

Generalized Gram–Schmidt process in exact arithmetic

Let $Z^{(0)} = [z_1^{(0)}, \dots, z_n^{(0)}] \in \mathbb{R}^{n\times n}$ be a nonsingular matrix. The non-generalized GS process (orthogonalization) provides decomposition of the matrix $Z^{(0)}$ in the form $Z^{(0)} = QR$ where $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix $Q^TQ = I$ and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix and can be seen as the Cholesky factor of the matrix $(Z^{(0)})^T Z^{(0)}$. Matrix Q is in this case computed by orthogonalization column of the matrix $Z^{(0)}$ (using orthogonal projections that employ the Euclidean inner product) against previously computed column vectors in Q.

The GGS process uses the non-standard inner-product induced by an SPD matrix $A \in \mathbb{R}^{n\times n}$. Special case $A = I$ corresponds to the (standard) Euclidean inner-product. Let us deal with identities for GGS, we have decomposition of the matrix $Z^{(0)}$ in the form $Z^{(0)} = ZU$ where $Z \in \mathbb{R}^{n\times n}$ has A-orthogonal columns such that $Z^TAZ = I$. Matrix $U \in \mathbb{R}^{n\times n}$ is an upper triangular matrix and can be seen as the Cholesky factor of the matrix $(Z^{(0)})^T AZ^{(0)}$ with the norm and minimum singular value

$$\begin{aligned} \|U\| &= \|A^{1/2}Z^{(0)}\|, \\ \sigma_n(U) &= \sigma_n(A^{1/2}Z^{(0)}) \end{aligned}$$

and for the condition number $\kappa(U)$ it holds $\kappa(U) = \kappa(A^{1/2}Z^{(0)})$. Due to the orthogonality relation $(A^{1/2}Z)^T(A^{1/2}Z) = I$, we have for the norm and minumum singular values of Z

$$\begin{aligned} \|Z\| &= \|A^{-1/2}\| = \|A^{-1}\|^{1/2}, \\ \sigma_n(Z) &= \lambda_n^{1/2}(A^{-1}) = \|A\|^{-1/2}. \end{aligned}$$

Since $Z = Z^{(0)}U^{-1}$ the product ZZ^T can be written as

$$ZZ^T = Z^{(0)}[(Z^{(0)})^T AZ^{(0)}]^{-1}(Z^{(0)})^T.$$

Then AZZ^T represents the oblique projector onto $\mathcal{R}(AZ^{(0)})$ and orthogonal to $\mathcal{R}(Z^{(0)})$. Similarly, ZZ^TA is the oblique projector onto $\mathcal{R}(Z^{(0)})$ and orthogonal to $R(AZ^{(0)})$. In the considered case we always have $ZZ^T = A^{-1}$ and $AZZ^T = ZZ^TA = I$.

In order to have all computations in the GGS process well defined, the matrix A has to be symmetric and positive definite, but note that this type of decomposition can be analyzed in the symmetric and indefinite case as well, see, e.g., [30].

2.1 The variants of the generalized Gram–Schmidt process

Let us summarize the main identities, matrices $Z = [z_1, \dots, z_n]$ and $U = [\alpha_{j,i}]$ produced by the GGS process satisfy the following identities

- $ZU = Z^{(0)}$
- $U^T U = (Z^{(0)})^T A Z^{(0)}$
- $Z^T A Z = I$
- $Z Z^T = A^{-1}$

We introduce also the notation for matrices with an extra subscript k (e.g., Z_k, U_k), that are leading principal submatrices corresponding to the row indices $i = 1, \dots, k$ and column indices $i = 1, \dots, k$ (of the matrices $Z^{(0)}, Z, U, \dots$). As well as for the leading principal submatrices of the product of the matrices we use notation $[\,\cdot\,]_k$, e.g, $[ZU]_k$. If we assume $Z^{(0)} = I$, all the previously mentioned identities hold also for $Z_k^{(0)}, Z_k, U_k$, and A_k. Then also for the Cholesky factor U of the matrix A it holds $U^{-1} = Z$ and moreover Z represents the inverse triangular factor in the factorization $A^{-1} = ZZ^T$ (this is also true for $Z^{(0)}$ in the upper triangular form).

Let us describe all the variants of the GGS process. The A-inner product of two vectors is here denoted by $\langle \cdot, \cdot \rangle_A$. The modified GGS process is given in Algorithm 1 in the so-called *left-looking* form, which means that the algorithm builds up the

Algorithm 1 Modified version of the GGS process (left-looking)

```
for $i := 1, \dots, n$ do
    for $j := 1, \dots, i-1$ do
        $\alpha_{j,i} := \langle z_i^{(j-1)}, z_j \rangle_A$
        $z_i^{(j)} := z_i^{(j-1)} - \alpha_{j,i} z_j$
    end for
    $\alpha_{i,i} := \langle z_i^{(i-1)}, z_i^{(i-1)} \rangle_A^{1/2}$
    $z_i := z_i^{(i-1)} / \alpha_{i,i}$
end for
```

column of the factor U by applying updates from left, using the already computed columns. Another variant of the modified GGS is the *right-looking* form that is given by Algorithm 2, which builds up the factor U by rows. The right-looking form of the

Algorithm 2 Modified version of the GGS process (right-looking)

```
for $j := 1, \dots, n$ do
    $\alpha_{j,j} := \langle z_j^{(j-1)}, z_j^{(j-1)} \rangle_A^{1/2}$
    $z_j := z_j^{(j-1)} / \alpha_{j,j}$
    for $i := j+1, \dots, n$ do
        $\alpha_{j,i} := \langle z_i^{(j-1)}, z_j \rangle_A$
        $z_i^{(j)} := z_i^{(j-1)} - \alpha_{j,i} z_j$
    end for
end for
```

algorithm may be beneficial if, for example, pivoting (column permutations in $Z^{(0)}$ in order to obtain U in a special form as we will see later) is needed. On the other hand, all the not yet orthogonalized vectors are projected in every major step j of the algorithm. This can be seen as a disadvantage for some computer architectures. In order to increase the amount of independent computations, Algorithm 1 can be reformulated to the scheme denoted here as Algorithm 3, which is also called the classical GGS process. The above mentioned variants of the GGS process are

Algorithm 3 Classical version of the GGS process

```
for $i := 1, \dots, n$ do
  for $j := 1, \dots, i-1$ do
    $\alpha_{j,i} := \langle z_i^{(j-1)}, z_j \rangle_A$
  end for
  for $j := 1, \dots, i-1$ do
    $z_i^{(j)} := z_i^{(j-1)} - \alpha_{j,i} z_j$
  end for
  $\alpha_{i,i} := \langle z_i^{(i-1)}, z_i^{(i-1)} \rangle_A^{1/2}$
  $z_i := z_i^{(i-1)} / \alpha_{i,i}$
end for
```

generally well known, but the GGS process is closely related to another possible algorithm. It has been published in [2] as an incomplete scheme for computing the factorized approximate inverse. Here, it is called the *AINV orthogonalization* and is given in Algorithm 4.

Algorithm 4 AINV orthogonalization process

```
for $i := 1, \dots, n$ do
  for $j := 1, \dots, i-1$ do
    $\alpha_{j,i} := \langle z_i^{(j-1)}, z_j^{(0)} \rangle_A / \alpha_{j,j}$
    $z_i^{(j)} := z_i^{(j-1)} - \alpha_{j,i} z_j$
  end for
  $\alpha_{i,i} := \langle z_i^{(i-1)}, z_i^{(i-1)} \rangle_A^{1/2}$
  $z_i := z_i^{(i-1)} / \alpha_{i,i}$
end for
```

There are several possibilities to compute the normalization coefficients (diagonal entries of the matrix U). The most natural formula

$$\alpha_{i,i} = \langle z_i^{(i-1)}, z_i^{(i-1)} \rangle_A^{1/2} \tag{2.1}$$

is common for all algorithms here. We call it the *stabilized* form (it has been proposed as a stabilization of the AINV algorithm in [1]). Another possible variant (originally proposed in [2]) is expressed by the formula motivated by the A-inner product in the classical GGS algorithm that is defined as

$$\alpha_{i,i} = \langle z_i^{(i-1)}, z_i^{(0)} \rangle_A^{1/2}, \tag{2.2}$$

we call it the *non-stabilized* form. The difference between these two approaches has been already discussed in [1]. The identity $U^T U = (Z^{(0)})^T A Z^{(0)}$ allows us also to compute diagonal entries in the same fashion as in the Cholesky factorization algorithm as follows

$$\alpha_{i,i} = \left(\langle z_i^{(0)}, z_i^{(0)} \rangle_A - \sum_{j=1}^{i-1} \alpha_{j,i}^2 \right)^{1/2}. \tag{2.3}$$

2.1.1 Pivoting in the GGS process

In the exact arithmetic, the pivoted GGS process, that is the process where column permutations of $Z^{(0)}$ are involved, produces matrices Z and U so that

- $ZU = Z^{(0)}P$
- $U^TU = (Z^{(0)}P)^TAZ^{(0)}P$
- $Z^TAZ = I$
- $ZZ^T = PA^{-1}P^T$

where P is a permutation matrix. The most common way of pivoting considers magnitudes of the diagonal entries in U. We deal in more detail with pivoting that involves column permutations of $Z^{(0)}$ such that for the entries in U then holds

$$\alpha_{1,1} \geq \alpha_{2,2} \geq \ldots \geq \alpha_{n,n} > 0, \tag{2.4}$$

$$\alpha_{i,i}^2 \geq \sum_{k=i}^{j} \alpha_{k,j}^2, \quad j = i+1, \ldots, n. \tag{2.5}$$

Note that both inequalities (2.4) and (2.5) also imply

$$\alpha_{j,j} > |\alpha_{j,k}|, \quad j = 1, \ldots, n, \quad k = j+1, \ldots, n. \tag{2.6}$$

The simplest way to obtain the Cholesky factor U in this form is to use the modified GGS in the right-looking form, see, Algorithm 2. When the whole kth row of the matrix U is computed and after projections of the vectors $z_i^{(k-1)}$, $i = k+1, \ldots, n$ onto A-orthogonal complement of z_k, we have the intermediate vectors $z_i^{(k)}$. Pivot ($(k+1, k+1)$ entry in the matrix U) then corresponds to the magnitude of the largest A-norm of the intermediate vectors such that

$$\|z_l^{(k)}\|_A = \max_{k+1 \leq i \leq n} \|z_i^{(k)}\|_A, \tag{2.7}$$

where l is the index corresponding to the $\|z_i^{(k)}\|_A$ largest in magnitude. Using formula (2.7) then leads to permutation of the column vectors $z_l^{(k)}$ and $z_{k+1}^{(k)}$.

It is also possible to propose another approach to get quantities $\|z_i^{(k)}\|_A$. We introduce a specific combination of the modified GGS algorithm in the left-looking form and the Cholesky factorization where quantities $\|z_i^{(k)}\|_A$ are computed similarly as it has been noted in (2.3). Formula

$$\|z_i^{(k)}\|_A^2 = \|z_i^{(0)}\|_A^2 - \sum_{j=1}^{k} \langle z_j, z_i^{(0)} \rangle_A^2 = \|z_i^{(k-1)}\|_A^2 - \langle z_k, z_i^{(0)} \rangle_A^2, \quad k < i \leq n \tag{2.8}$$

shows how the computed vector z_k can be used to update the A-norms of the vectors, which have not been A-orthogonalized yet. Pivot entry can be then chosen as for right-looking modified GGS using (2.7). In this way, especially for $Z^{(0)} = I$, the considered pivoting is easy and cheap to compute. Pivoting (that is cheap to compute) has been already used, e.g., in [27]. Using the recursive (as defined rightmost in formula (2.8)) computation of $\|z_i^{(k)}\|_A$; quantities $\|z_i^{(k-1)}\|_A^2$ are known from previous major step, they are updated by squares of the entries of the vector Az_k (for $Z^{(0)} = I$).

The Algorithm 1 extended by the use of (2.8) that we call the Cholesky-based pivoting, is given by Algorithm 5. The entries of the vector

$$d^{(i-1)} = [d_1^{(0)}, d_2^{(1)}, \ldots, d_{i-1}^{(i-2)}, d_i^{(i-1)}, d_{i+1}^{(i-1)}, \ldots, d_n^{(i-1)}]$$

Algorithm 5 Modified version of the GGS process (left-looking) employing the Cholesky-based pivoting

```
$P^{(0)} = I$
$d^{(0)} := (d_1^{(0)}, d_2^{(0)}, \dots, d_n^{(0)}) = (a_{1,1}, a_{2,2}, \dots, a_{n,n})$
for $i := 1, \dots, n-1$ do
  if $i > 1$ then
    for $j := i, \dots, n$ do
      $d_j^{(i-1)} = d_j^{(i-2)} - \langle z_{i-1}, z_j^{(0)} \rangle_A^2$
    end for
  end if
  $l = \underset{i \le j \le n}{\mathrm{argmax}}(d_j^{(i)})$
  $Z^{(0)} = \mathrm{swap\_columns}_{i,l}(Z^{(0)})$
  $P^{(i)} = \mathrm{swap\_columns}_{i,l}(P^{(i-1)})$
  $\mathrm{swap\_entries}_{i,l}(d^{(i)})$
  for $j := 1, \dots, i-1$ do
    $\alpha_{j,i} := \langle z_i^{(j-1)}, z_j \rangle_A$
    $z_i^{(j)} := z_i^{(j-1)} - \alpha_{j,i} z_j$
  end for
  $\alpha_{i,i} = \langle z_i^{(i-1)}, z_i^{(i-1)} \rangle_A^{1/2}$
  $z_i = z_i^{(i-1)} / \alpha_{i,i}$
end for
$P = P^{(n-1)}$
```

plays the same role as quantities $\|z_i^{(k)}\|_A^2$ in formula (2.8), in this way are also updated. Positions in the vector $d^{(i-1)}$ correspond to the column vectors in $Z^{(0)}P^{(i-1)}$, where $P^{(i-1)}$ is the $i-1$ iteration to P. The formula $l = \underset{i \le j \le n}{\mathrm{argmax}}(d_j^{(i-1)})$ finds the index of the pivot entry having the largest magnitude in the ith step (satisfying (2.7)). The *swap* operations permute the corresponding entries or columns with respect to the position of the pivot. Moreover, *swap* operations interchange also the indices of the permuting quantities. Note that $Z^{(0)}$ in Algorithm 5 is rewritten in every step. The permutation introduced in Algorithm 5 can be interpreted as running the generalized Gram–Schmidt process with $Z^{(0)}$ that is equal to the corresponding permutation matrix P instead of the standard choice $Z^{(0)} = I$ or equivalently as running the generalized Gram–Schmidt process where $Z^{(0)} = I$ with the already permuted matrix $A' = P^TAP$. The permutation matrix P is not a priori known. The column permutations in $P^{(i-1)}$ are performed *on-the-fly* by employing (2.8) and (2.7). Consequently, for $P \neq I$, Z does not have the upper triangular form even for $Z^{(0)} = I$. In this case we have $P^TZ = U^{-1}$.

Chapter 3

Generalized Gram–Schmidt in finite precision arithmetic

In order to distinguish between the quantities computed in finite precision arithmetic and their exact arithmetic counterparts, we denote the quantities computed in finite precision arithmetic by an extra upper bar (e.g., $\bar{Z}, \bar{U}$).

For initial vector basis $Z^{(0)}$ in a general form it has been shown in [31] that the loss of A-orthogonality $\|\bar{Z}^T A\bar{Z} - I\|$ can be significantly different for various numerical implementations. Despite this fact, it has been also shown in [31] that their left residuals $\|\bar{Z}\bar{U} - Z^{(0)}\|$ is approximately of the same order of magnitude.

From now, we will deal with the special case, where the initial vector basis has the form $Z^{(0)} = I$. Moreover, some proofs in this chapter could be done only by assuming that the GGS algorithm uses pivoting such that it holds (2.4) and (2.5) for the entries of $\bar{U}$.

In this chapter we consider component-wise error bounds for the left residual $|\bar{Z}\bar{U} - I|$, and the right residual $|\bar{U}\bar{Z} - I|$, error bounds in the inverse $|\bar{U}^{-1} - \bar{Z}|$, and the error bounds for Cholesky factorization $|A - \bar{U}^T\bar{U}|$.

All the variants of the GGS process that are described in Section 2.1 differ in the definition of the A-orthogonalization coefficients $\alpha_{j,i}$. Therefore, they give in finite precision arithmetic different resulting matrices $\bar{U} = [\bar{\alpha}_{j,i}]$.

On the other hand, the matrix $\bar{Z}$ (obtained by inverting the matrix $\bar{U}$ in finite precision arithmetic) is computed in the same way for all algorithms. The algorithmic scheme to compute $\bar{Z}$ corresponds numerically to the left-looking scheme (so-called j-version) [25] (or on the numerically equivalent so-called k-version [25] for right-looking modified GGS) for computing inverse $X = \bar{Z}$ of the upper triangular matrix $\bar{U}$ from the equation $X\bar{U} = I$. Therefore, some proofs can be done in a common way for more variants of the GGS process.

Assume the following *bordering notation* using the matrices

$$\bar{U}_k = \begin{pmatrix} \bar{U}_{k-1} & \bar{u}_k \\ 0 & \bar{\alpha}_{k,k} \end{pmatrix}, \quad \bar{Z}_k = \begin{pmatrix} \bar{Z}_{k-1} & \bar{w}_k \\ 0 & \bar{\beta}_{k,k} \end{pmatrix}. \tag{3.1}$$

The last column vector in $\bar{Z}_k$ is computed in finite precision arithmetic using the already computed block $\bar{Z}_{k-1}$ and the last column vector in $\bar{U}_k$. Therefore, for the quantities $\bar{w}_k$ and $\bar{\beta}_{k,k}$ we can write for $k = 1, \dots, n$:

$$\bar{\beta}_{k,k} = \frac{1}{\bar{\alpha}_{k,k}} + \delta\bar{\beta}_{k,k}, \qquad |\delta\bar{\beta}_{k,k}| \le \frac{u}{\bar{\alpha}_{k,k}}, \tag{3.2}$$

and for $k = 2, \dots, n$:

$$\bar{w}_k = -\bar{Z}_{k-1}\bar{u}_k\bar{\beta}_{k,k} + \delta w_k, \qquad |\delta w_k| \le \mathcal{O}(k)u|\bar{Z}_{k-1}||\bar{u}_k|\bar{\beta}_{k,k}. \tag{3.3}$$

In order to be consistent with the already introduced notation, naturally, it holds

$$\bar{z}_k = \begin{pmatrix} \bar{w}_k \\ \bar{\beta}_{k,k} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} n-k.$$

3.1 Right residuals

Lemma 3.1. *Let $\bar{Z}_k$ and $\bar{U}_k$ be approximations to the factors Z_k and U_k, respectively, that are computed by Algorithm 5 (or any other GGS algorithm without iterative refinement with pivoting such that it holds (2.4) and (2.5) for the entries of $\bar{U}$) in finite precision arithmetic with the unit roundoff u. Let $\bar{D}_k$ be the diagonal matrix having the same diagonal as $\bar{U}_k$. Then*

$$\bar{U}_k \bar{Z}_k = I_k + \Delta E_k^{(2)}, \tag{3.4}$$

where the matrix of right residuals $\Delta E_k^{(2)}$ can be bounded by

$$|\Delta E_k^{(2)}| \leq \mathcal{O}(k) u |\bar{U}_k||\bar{Z}_k||\bar{U}_k|\bar{D}_k^{-1}. \tag{3.5}$$

Proof. The proof is by using induction. For $k = 1$, it holds trivially, similarly as for the left residuals. For $k = 2, \ldots, n$, we using (3.1) get

$$\bar{U}_k \bar{Z}_k - I_k = \begin{pmatrix} \bar{U}_{k-1} & \bar{u}_k \\ 0 & \bar{\alpha}_{k,k} \end{pmatrix} \begin{pmatrix} \bar{Z}_{k-1} & \bar{w}_k \\ 0 & \bar{\beta}_{k,k} \end{pmatrix} - I_k =$$

$$= \begin{pmatrix} \bar{U}_{k-1}\bar{Z}_{k-1} - I_{k-1} & \bar{U}_{k-1}\bar{w}_k + \bar{u}_k \bar{\beta}_{k,k} \\ 0 & \bar{\alpha}_{k,k}\bar{\beta}_{k,k} - 1 \end{pmatrix}.$$

Assume the results (3.5) is true for matrices of the order $k-1$. After some manipulations and using (3.2) and (3.3), we obtain

$$\begin{aligned} \bar{U}_{k-1}\bar{w}_k + \bar{u}_k\bar{\beta}_{k,k} &= \bar{U}_{k-1}(-\bar{Z}_{k-1}\bar{u}_k\bar{\beta}_{k,k} + \delta w_k) + \bar{u}_k\bar{\beta}_{k,k} \\ &= (I_{k-1} - \bar{U}_{k-1}\bar{Z}_{k-1})\bar{u}_k\bar{\beta}_{k,k} + \bar{U}_{k-1}\delta w_k \\ &\leq |I_{k-1} - \bar{U}_{k-1}\bar{Z}_{k-1}||\bar{u}_k|\bar{\beta}_{k,k} + \mathcal{O}(k)u|\bar{U}_{k-1}||\bar{Z}_{k-1}||\bar{u}_k|\bar{\beta}_{k,k} \\ &\leq \mathcal{O}(k)u|\bar{U}_{k-1}||\bar{Z}_{k-1}||\bar{U}_{k-1}|\bar{D}_{k-1}^{-1}|\bar{u}_k|\bar{\beta}_{k,k} \\ &\quad + \mathcal{O}(k)u|\bar{U}_{k-1}||\bar{Z}_{k-1}||\bar{u}_k|\bar{\beta}_{k,k} \\ &= \mathcal{O}(k)u|\bar{U}_{k-1}||\bar{Z}_{k-1}\bar{D}_{k-1}|\underbrace{|\bar{D}_{k-1}^{-1}\bar{U}_{k-1}|}_{\leq \mathcal{O}(1)}|\bar{D}_{k-1}^{-1}\bar{u}_k|\bar{\beta}_{k,k} \\ &\quad + \mathcal{O}(k)u|\bar{U}_{k-1}||\bar{Z}_{k-1}||\bar{u}_k|\bar{\beta}_{k,k} \\ &= \mathcal{O}(k)u|\bar{U}_{k-1}||\bar{Z}_{k-1}||\bar{u}_k|\bar{\beta}_{k,k}. \end{aligned} \tag{3.6}$$

Note that we assume the algorithm with pivoting such that it holds (2.4) and (2.5) for the entries of $\bar{U}$. Therefore we have $|\bar{D}_{k-1}^{-1}\bar{U}_{k-1}| \leq 1 \leq \mathcal{O}(1)$. Inequality (3.6) holds for $k = 2, \ldots, n$ and the result (3.5) is proved. □

3.2 Norm-wise vs. component-wise error bounds

Component-wise error bounds may be, in general, tighter than norm-wise ones, it has been already mentioned in Chapter 1. Let us deal with this problem more in details.

3.2.1 Skeel's condition number

The difference between norm-wise and component-wise error bounds can be explained via the condition number introduced by Skeel [32].

Definition 3.1. *Let $B \in \mathbb{R}^{n\times n}$ be a nonsingular matrix. Then*

$$\mathrm{cond}(B) \equiv \| |B^{-1}||B| \|_\infty \tag{3.7}$$

is the Skeel's condition number.

From Definition 3.1, it is easy to see that cond(B) is invariant under row scaling of B. It means $\mathrm{cond}(B) = \mathrm{cond}(D_s B)$ for all the diagonal matrices D_s. It is also straightforward to show that

$$\mathrm{cond}(B) \equiv \min_{D_R \text{ diagonal}} \left(\kappa_\infty(D_R B)\right). \tag{3.8}$$

Remark 3.1. *Note that, in general, we have*

$$\mathrm{cond}(B) \neq \mathrm{cond}(B^{-1}).$$

For the Skeel's condition number cond$(\cdot)$, condition number induced by infinity norm $\kappa_\infty(\cdot)$, and condition number (with respect singular values) $\kappa(\cdot)$ we can put down trivial inequality

$$\mathrm{cond}(B) \leq \kappa_\infty(B) \leq n\,\kappa(B). \tag{3.9}$$

The optimal row scaling matrix in (3.8) is given by $[D_R]_{j,j} = \sum_{i=1}^n [|B|]_{j,i}$. Thus the sum of the absolute values of the entries in the rows of the scaled matrix $D_R B$ are equal to one. In [11] can be found following inequality

$$\kappa_\infty(B) \leq \kappa_\infty(D_R)\mathrm{cond}(B), \tag{3.10}$$

that can be also rewritten using spectral norms as

$$\kappa(B) \leq n\,\kappa(D_R)\mathrm{cond}(B). \tag{3.11}$$

3.2.2 Right residuals from the point of view of the Skeel's condition number

The right hand side of (3.5) can be rewritten as

$$|\bar{U}||\bar{Z}||\bar{U}|\bar{D}^{-1} = \bar{D}|\bar{D}^{-1}\bar{U}||\bar{Z}||\bar{U}|\bar{D}^{-1}. \tag{3.12}$$

Using $\|\bar{D}^{-1}\bar{U}\| \leq \mathcal{O}(1)$ and the Cauchy–Schwarz inequality, we get

$$\| |\bar{U}||\bar{Z}||\bar{U}|\bar{D}^{-1} \| \leq \mathcal{O}(n^{1/2})\kappa(\bar{D})\mathrm{cond}(\bar{U}). \tag{3.13}$$

The inequality (3.11) with $\bar{U}$ substituted for B and $\bar{D}$ substituted for D_R is similar to (3.13). There are different constants ($\mathcal{O}(n)$ vs. $\mathcal{O}(n^{1/2})$) because D_R and $\bar{D}$ differs at most by the factor $\mathcal{O}(n^{1/2})$. Both inequalities (3.11) and (3.13) were obtained via the Cauchy–Schwarz inequality, therefore the right hand sides may tend to overestimate. Relation between $\| |\bar{U}||\bar{Z}||\bar{U}|\bar{D}^{-1} \|$ and $\kappa(\bar{U})$ remains an open problem at present time.

3.3 Summary of the error bounds

The error bounds introduced here represent a complement (or the component-wise counterparts) of the error bounds developed in [31]. In particular, in thesis, we are focused on the GGS algorithm with pivoting. We summarize all the developed error bounds in a compact form in Table 3.1.

Table 3.1: Comparison of upper bounds for the left residuals, right residuals, and the Cholesky factorization error for different orthogonalization schemes

non-pivoted algorithms			
	$\lvert\bar{Z}\bar{U} - I\rvert$	$\lvert\bar{U}\bar{Z} - I\rvert$	$\lvert A - \bar{U}^T\bar{U}\rvert$
classical GGS (non-stabilized)	$\leq \mathcal{O}(n)u\lvert\bar{Z}\rvert\lvert\bar{U}\rvert$	$\leq \mathcal{O}(n)u\lvert\bar{U}\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert\lvert\bar{Z}\rvert$	$\leq \mathcal{O}(n)u(\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert + (\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert)^T)$
classical GGS (stabilized)	$\leq \mathcal{O}(n)u\lvert\bar{Z}\rvert\lvert\bar{U}\rvert$	$\leq \mathcal{O}(n)u\lvert\bar{U}\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert\lvert\bar{Z}\rvert$	$\lessapprox \mathcal{O}(n)u(\lvert\bar{Z}\rvert\lvert\bar{U}\rvert)^T\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert$
modified GGS	$\leq \mathcal{O}(n)u\lvert\bar{Z}\rvert\lvert\bar{U}\rvert$	$\leq \mathcal{O}(n)u\lvert\bar{U}\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert\lvert\bar{Z}\rvert$	$\leq \mathcal{O}(n)u(\lvert\bar{Z}\rvert\lvert\bar{U}\rvert)^T\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert$
pivoted algorithms			
	$\lvert\bar{P}^T\bar{Z}\bar{U} - I\rvert$	$\lvert\bar{U}\bar{P}^T\bar{Z} - I\rvert$	$\lvert\bar{P}^T A\bar{P} - \bar{U}^T\bar{U}\rvert$
classical GGS (non-stabilized)	$\leq \mathcal{O}(n)u\lvert\bar{P}^T\bar{Z}\rvert\lvert\bar{U}\rvert$	$\leq \mathcal{O}(n)u\lvert\bar{U}\rvert\lvert\bar{P}^T\bar{Z}\rvert\lvert\bar{U}\rvert\bar{D}^{-1}$	$\leq \mathcal{O}(n)u\left(\lvert\bar{U}^T\rvert\lvert\bar{Z}^T\rvert\lvert A\rvert\bar{P} + \bar{P}^T\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert\right)$
classical GGS (stabilized)	$\leq \mathcal{O}(n)u\lvert\bar{P}^T\bar{Z}\rvert\lvert\bar{U}\rvert$	$\leq \mathcal{O}(n)u\lvert\bar{U}\rvert\lvert\bar{P}^T\bar{Z}\rvert\lvert\bar{U}\rvert\bar{D}^{-1}$	$\lessapprox \mathcal{O}(n)u\left(\lvert\bar{U}^T\rvert\lvert\bar{Z}^T\rvert\lvert A\rvert\bar{P} + \bar{P}^T\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert + \mathrm{diag}(\lvert\bar{U}^T\rvert\lvert\bar{Z}^T\rvert\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert)\right)$
modified GGS	$\leq \mathcal{O}(n)u\lvert\bar{P}^T\bar{Z}\rvert\lvert\bar{U}\rvert$	$\leq \mathcal{O}(n)u\lvert\bar{U}\rvert\lvert\bar{P}^T\bar{Z}\rvert\lvert\bar{U}\rvert\bar{D}^{-1}$	$\leq \mathcal{O}(n)u\lvert\bar{U}^T\rvert\lvert\bar{Z}^T\rvert\lvert A\rvert\lvert\bar{Z}\rvert\lvert\bar{U}\rvert$

Chapter 4

Generalized Gram–Schmidt process as an incomplete scheme

This chapter is devoted to the computation of the *incomplete* GGS process. We assume that the incomplete GGS process may provide good approximations that can be used to precondition an iterative method. As a theoretical motivation, we consider the *complete* GGS process, that is, the process computed in the floating-point arithmetic without any dropping.

In particular, we will deal with value-based dropping strategies where the entries in the computed factors or intermediate quantities that exceed a prescribed threshold are *dropped*. It is clear that the success of such approaches often depends on an ability to find a suitable threshold. This can be highly problem dependent for dropping based on magnitudes even if the entries are dropped comparing their magnitudes relatively to well-chosen quantities connected to the matrix.

The reason is clear: Accuracy of the decomposition is related to global properties of the given matrix, e.g., to the singular values. Matrices can be seen as discrete operators, where the eigenvalues λ_i and the singular values σ_i are continuous functions of the coefficients of the matrix, but the dependency is strongly non-linear. Therefore, simple dropping rules that do not take into account global properties of the given matrix related to the accuracy of the decomposition can only very hardly provide robust preconditioners in all the cases.

Triangular decompositions or inversion of a triangular matrix allow us to use the bordering scheme (depend on given algorithm) where it is easy to see that the intermediate rows/columns are affected (in terms of rounding errors) only by one block of the matrix. Therefore, rounding errors in such rows/columns are also affected only by singular values that correspond to the related block of the matrix (the interlacing theorem for singular values [39]), not to the whole matrix.

Incomplete algorithms are based on dropping rules that are, in particular, often motivated mainly by heuristics although in some cases there exists a reasonable theoretical understanding of the incomplete process. Based on the new theoretical considerations (introduced previously) and also numerical behavior we introduce a new dropping scheme that takes into account properties of finite precision arithmetic and also a global information from the input data (matrix A).

4.1 Motivation

Factorized approximate inverse preconditioners are based on representing the inverse A^{-1} via the incomplete decomposition $\widetilde{Z}\widetilde{Z}^T$. In our case, $\widetilde{Z}$ is in the upper triangular form. Our system of linear algebraic equations (1.4) can be rewritten in the symmetrically preconditioned form as

$$\widetilde{Z}^T A\widetilde{Z}y = \widetilde{Z}^T b, \qquad x = \widetilde{Z}y. \tag{4.1}$$

First, let us summarize properties needed to obtain an incomplete decomposition that could provide a good preconditioner. Practical preconditioners have to be *sparse*. This is achievable for standard incomplete decompositions, but it is more difficult for the approximate inverse decompositions since they may fill very quickly. Further, preconditioners should be as *accurate* as possible. Therefore, the quality of the approximate inverse preconditioning can be assessed by following indicators:

(i) fill-in in the matrix $\widetilde{Z}$ (represented by the *number of nonzeros* denoted as $\mathrm{nnz}(\widetilde{Z})$),

(ii) the loss of the A-orthogonality among column vectors of $\widetilde{Z}$ measured by $\|\widetilde{Z}^T A\widetilde{Z} - I\|$.

Fill-in in the matrix $\widetilde{Z}$ determines to the cost of the preconditioner computation and its application. A measure of the quality of the computed preconditioner that we will call stability is represented by $\|\widetilde{Z}^T A\widetilde{Z} - I\|$ or $\|U\widetilde{Z} - I\|$, and it is usually in contradiction with the sparsity of the preconditioner. Therefore, simultaneous minimization of these indicators is not an easy task for a general matrix A. The goal is to find a *balance* between them.

Remark 4.1. *Note that the loss of A-orthogonality $\|\widetilde{Z}^T A\widetilde{Z} - I\|$ as well as the norm $\|A - (\widetilde{Z}\widetilde{Z}^T)^{-1}\|$ plays similar roles as the stability of the preconditioner $\|I - \widetilde{L}^{-1}A\widetilde{L}^{-T}\|$ and the accuracy of the preconditioner $\|A - \widetilde{L}\widetilde{L}^T\|$, respectively, in the incomplete Cholesky factorization, [12].*

4.2 Incomplete algorithms, error bounds and dropping strategies

The incomplete algorithm relaxes the decomposition by dropping small entries (small in some well-defined sense). The choice of a specific strategy depends on properties of the individual algorithm possibly taking into account also the target computer architecture. It has been shown in [2] that such strategy used in the context of the generalized Gram–Schmidt process may produce a good and competitive preconditioner. Nevertheless, the choice of the optimal drop tolerance may be difficult. Up to now, there has not yet been proposed a better way to find it in general cases than by *trial-and-error*.

In the ideal case, we may easily detect and drop those entries that do not significantly contribute to the accuracy of computed factors. The analysis in this work motivates us to find tools useful for decision which nonzero fill-in entries do not significantly contribute to this accuracy.

We have shown (even in the presence of rounding errors) that the considered variants of the GGS algorithm produce the matrices $\bar{Z}$ and $\bar{U}$ that are good approximations to Z and U, respectively. But such computation is inevitably time consuming because it leads to rather dense factors. In Chapter 3, we have dealt with error bounds for GGS in finite precision arithmetic. Our dropping rules should reflect these theoretical error bounds.

4.3 Construction of the incomplete scheme

4.3.1 Summary of theory and numerical experiments

In thesis, we have shown on test problems that numerical properties of some variants of GGS differ accordingly to the error bounds developed in Chapter 3. From (4.1), it is clear that a dropping technique should be based on the error bound for the loss of A-orthogonality or similarly on the error bound for $|U\bar{Z} - I|$ that also reflects accuracy of the approximation of the inverse of U from the right. This is also related to the error bound for the right residuals. Our error bounds are tight, but especially for the right residuals $|\bar{U}\bar{Z} - I|$ it seems (based partially on the theory as well as on observations) that they can be expressed in the terms of singular values of the matrix A (or similarly in the terms of the norms of the computed matrices $\bar{Z}$ and $\bar{U}$) as $\kappa^{1/2}(A) \approx \|\bar{U}\|\|\bar{Z}\|$. Therefore, in the following text we assume that the error in decomposition (3.4) is given by

$$|\Delta E_k^{(2)}| \leq \mathcal{O}(k)u\|\bar{U}_k\|\|\bar{Z}_k\|. \tag{4.2}$$

4.3.2 Adaptive dropping by a posteriori filtering

In order to present the main idea of the a posteriori filtering, we do not explicitly deal with the pivoted algorithm because the pivoted algorithm represents a non-pivoted algorithm that uses the already permuted matrix $A' = P^TAP$ as presented in subsection 2.1.1. On the other hand, we emphasize technical differences, especially treatment of the sparsity pattern.

Although the GGS process in finite precision arithmetic provides approximations $\bar{Z}$ and $\bar{U}$ of the exact matrices Z and U, respectively, preconditioning of the linear system (1.4) by $\bar{Z}$ is related more to the direct method then to the preconditioning. Our goal is to construct a dropping strategy that reflects in some sense properties of the finite precision arithmetic. For the quantities that are influenced by dropping we also introduce notation with an extra tilde (e.g., $\widetilde{Z}$) in order to be consistent with the notation used in Section 4.1. Assume that some entries in the already orthogonalized vectors are discarded in the following way

$$z_k^{(k-1)} \circ (\mathbb{1} - s_k) = z_k^{(k-1)} - z_k^{(k-1)} \circ s_k = \widetilde{z}_k^{(k-1)},$$

where the symbol "$\circ$" denotes the Hadamard (entry-wise) product, $\mathbb{1} = \sum_{i=1}^n e_i$, and where $s_k \in \{0,1\}^n$ is a $(0,1)$-vector that specifies which entries will be discarded. Moreover, such vectors are normalized as follows

$$\widetilde{z}_k = \frac{\widetilde{z}_k^{(k-1)}}{\|\widetilde{z}_k^{(k-1)}\|_A} = \frac{z_k^{(k-1)} \circ (\mathbb{1} - s_k)}{\|z_k^{(k-1)} \circ (\mathbb{1} - s_k)\|_A}. \tag{4.3}$$

Assume that the GGS process runs in the arithmetic with the unit roundoff τ_k. Then using (4.3) we can rewrite the last column of the equality (4.2) using quantities with tilde as

$$\widetilde{U}_k[\widetilde{z}_k]_{1:k} - e_k = [\Delta E_k^{(2)}]_{1:k,k} - \widetilde{U}_k \frac{[z_k^{(k-1)} \circ s_k]_{1:k}}{\|z_k^{(k-1)} \circ (\mathbb{1} - s_k)\|_A}, \tag{4.4}$$

with

$$[|\Delta E_k^{(2)}|]_{1:k,k} \leq \mathcal{O}(k)\tau_k\|\widetilde{U}_k\|\|\widetilde{z}_k\|, \tag{4.5}$$

where the notation $[\,\cdot\,]_{1:k}$ for the vectors has the same meaning as for matrices (here it denotes restriction to the first k entries).

Let us assume that the magnitude of the error term that arises from dropping (second term on the right hand side (4.4)) is at most as large as the magnitude of the rounding errors inherent to the GGS process (first term on the right hand side (4.4)). Taking both error terms in the norms, using (4.5), and assuming that both error terms are not larger than a chosen tolerance τ, $0 \leq \tau \leq 1$, we have

$$\mathcal{O}(k^{3/2})\|\widetilde{U}_k\| \frac{\|z_k^{(k-1)} \circ s_k\|_\infty}{\|z_k^{(k-1)} \circ (\mathbb{1} - s_k)\|_A} \lessapprox \mathcal{O}(k^{3/2})\tau_k \|\widetilde{U}_k\| \|\widetilde{z}_k\|_\infty \leq \tau. \tag{4.6}$$

Dividing (4.6) by $\mathcal{O}(k^{3/2})\|\widetilde{U}_k\|\|\widetilde{z}_k\|_\infty$ and using $\kappa(\widetilde{U}_k) \lessapprox \mathcal{O}(k^{3/2})\|\widetilde{U}_k\|\|\widetilde{z}_k\|_\infty$ and (4.3), we get

$$\frac{\|z_k^{(k-1)} \circ s_k\|_\infty}{\|z_k^{(k-1)} \circ (\mathbb{1} - s_k)\|_\infty} \lessapprox \tau_k \leq \frac{\tau}{\kappa(\widetilde{U}_k)}. \tag{4.7}$$

Let us consider s_k in more detail. Our goal is to find a sparsity pattern of the vector z_k (represented by the entries equal to *one* in corresponding positions in the s_k vector) such that discarding all the nonzero entries outside this sparsity pattern does not significantly increase the right residuals (errors stay of the order $\mathcal{O}(k)\tau$). The largest entries (in terms of their magnitudes) have to be preserved, this leads to the identity

$$\|z_k^{(k-1)} \circ (\mathbb{1} - s_k)\|_\infty = \|z_k^{(k-1)}\|_\infty. \tag{4.8}$$

Therefore, (4.7) can be rewritten by using (4.8) as

$$\frac{\|z_k^{(k-1)} \circ s_k\|_\infty}{\|z_k^{(k-1)}\|_\infty} \lessapprox \tau_k \leq \frac{\tau}{\kappa(\widetilde{U}_k)}. \tag{4.9}$$

The sparsity pattern represented by the entries equal to 1 at the corresponding positions in s_k can be obtained from the inequality (4.9). Let us set the entries at the corresponding positions to 0 or 1 so that the inequality (4.9) holds for all its entries of the vector $z_k^{(k-1)}$ (except for the kth entry in the vector $z_k^{(k-1)}$ for non-pivoted algorithms or except for the kth entry in the vector $(\widetilde{P}^{(k-1)})^T z_k^{(k-1)}$ for pivoted algorithms). Then it is clear how the large entries (with respect to their magnitudes) can be dropped for a given τ.

Remark 4.2. *The diagonal entries of $\widetilde{Z}$ for the non-pivoted case and the diagonal entries of $\widetilde{P}^T\widetilde{Z}$ for the pivoted case has to be involved to the sparsity pattern in order to obtain non-singular preconditioner.*

This new dropping technique based on monitoring the condition number of $\widetilde{U}_k$ will be called the *a posteriori filtering.* Further, the singular values interlacing property $\kappa(\widetilde{U}_1) \leq \kappa(\widetilde{U}_2) \leq \cdots \leq \kappa(\widetilde{U}_k) \leq \cdots \leq \kappa(\widetilde{U}_n)$ [39] implies that the sequence of drop tolerances τ_k is non-increasing. Typically, the relative error $\|\widetilde{z}_k^{(k-1)} \circ s_k\|_\infty / \|\widetilde{z}_k^{(k-1)}\|_\infty$ decreases as $\kappa(\widetilde{U}_k)$ increases. Note that, the proposed dropping strategy does not depend on the conditioning of the whole problem $\kappa(U)$, but only on the local condition numbers $\kappa(\widetilde{U}_k) \approx U_k$. Therefore, a natural practical strategy is to keep the increase in the sequence of the local condition numbers $\kappa(\widetilde{U}_k)$ as small as possible.

4.4 Numerical aspects of a posteriori filtering

Motivation of our dropping strategy is clear for the algorithms with pivoting. Nevertheless, for comparison purposes we will use this technique also for the non-pivoted algorithms.

As we can see from (4.9), the a posteriori filtering may allow to drop relatively large entries when the local condition numbers reflecting the first terms of the sequence $\kappa(\widetilde{U}_1), \ldots, \kappa(\widetilde{U}_k)$ is small. This fact implies the need to minimize and also monitor the local condition numbers $\kappa(\widetilde{U}_k)$.

4.4.1 Minimizing of the local condition numbers

It is necessary to mention that the local condition numbers $\kappa(\widetilde{U}_k)$ can not be minimized independently of each other (the interlacing theorem for singular values). We know that $\kappa(U_1) = 1$ and $\kappa(\widetilde{U}_n) \approx \kappa^{1/2}(A)$. Shape of the non-increasing curve of the local condition numbers $\kappa(\widetilde{U}_k)$ with increasing k can be modified using a symmetric reordering $A \mapsto \Pi^T A \Pi$ with a permutation matrix Π. The simplest and intuitive assumption leads to minimizing the functional

$$\psi = \sum_{k=1}^{n} \kappa(\widetilde{U}_k), \tag{4.10}$$

and it is motivated by the greedy approach to drop as much as possible as long as possible.

We have already described pivoting in the GGS process. Of course, the considered pivoting can be seen as a process that tries to keep the local condition numbers $\kappa(\widetilde{U}_k)$ small as long as possible, i.e., it approximately minimizes the functional (4.10). In the other words this pivoting implies (2.4) and (2.5), therefore, the dominant information (in some sense associated to large singular values) from A is processed first.

Even in presence of pivoting, the curve of local condition numbers ($\kappa(\widetilde{U}_k)$ as a function of k) may locally (between two consecutive major steps) grow very fast if there is a gap in the decay of the singular values of $\widetilde{U}$.

It is a well known fact that scaling of the matrix is one of the simplest preprocessing technique that may improve conditioning of the matrix and it also may force more uniform distribution of the eigenvalues. Here, we will use the scaling introduced by Lin and Moré in [28]. Based on experimental results, we propose to use it iteratively (similarly as in MC77 [41]) as shown in Algorithm 6.

Algorithm 6 Iterative computation of the scaling by Lin and Moré

```
$A^{(0)} = A = [a_1^{(0)} a_2^{(0)} \ldots a_n^{(0)}]$
$D^{(0)} = I_n$
for $k = 1, 2, \ldots$   until   $(| \|a_i^{(k)}\| - 1| \le \theta,\ \forall i)$ or $(k > k_{\mathtt{max}})$ do
    $[D]_{i,i} = \|a_i^{(k)}\|^{1/2}, \quad i = 1, 2, \ldots, n$
    $A^{(k+1)} = D^{-1} A^{(k)} D^{-1}$
    $D^{(k+1)} = D^{(k)} D$
end for
```

Iterative scaling by Lin and Moré (or simply iterative Lin–Moré scaling) transforms the original matrix A into the form

$$A^{(D)} \equiv (D^{(k)})^{-1} A (D^{(k)})^{-1},$$

where all the columns (and also rows) of the matrix $A^{(D)}$ have approximately unit Euclidean norms.

4.4.2 Monitoring of the local condition numbers

Our method focuses on computation of the approximate inverse preconditioning that should stay cheap. Therefore, expensive computations of the singular value decomposition in order to get local condition numbers $\kappa(\widetilde{U}_k)$ in every major step can not be the method of choice. For the pivoted algorithms we can deliver the lower and also the realistic estimate of local condition numbers $\kappa(\widetilde{U}_k)$.

From the properties of the eigenvalues and singular values, we have that $\kappa_S(\widetilde{U}_k) \leq \kappa(\widetilde{U}_k)$, where

$$\kappa_S(\widetilde{U}_k) \equiv \frac{\max\limits_{1\leq i\leq k} [\widetilde{U}_k]_{i,i}}{\min\limits_{1\leq i\leq k} [\widetilde{U}_k]_{i,i}} \approx \frac{\widetilde{\alpha}_{1,1}}{\widetilde{\alpha}_{k,k}} \tag{4.11}$$

is the spectral condition number. Almost all diagonal entries in $\widetilde{U}_k$ hold analogous inequalities as formula (2.4) for the diagonal entries in U computed by GGS in exact arithmetic. Therefore, $\kappa_S(\widetilde{U}_k)$ is non-decreasing with k. The accuracy of the estimate $\kappa(\widetilde{U}_k)$ by $\kappa_S(\widetilde{U}_k)$ depends on distance from normality. Note that several ways of measuring the distance from normality can be found in paragraph 48 of [40]. Therefore, an optimistic estimate (lower bound) of $\kappa(\widetilde{U}_k)$ can be based on the spectral condition number $\kappa_S(\widetilde{U}_k)$.

A more realistic estimation of $\kappa(\widetilde{U}_k)$ (also for non-pivoted algorithms) can be based on the incremental condition estimation [6, 7, 8] or on more recent improvement of this method [14].

4.5 Incomplete GGS based on the a posteriori filtering

Up to now, we have dealt with the GGS process by running the "complete" algorithm in exact and also in finite precision arithmetic. The dropping rule developed in Section 4.3 in some sense obliterates the differences between the considered "incomplete" GGS process and the GGS process in finite precision arithmetic because the quantity τ_k can be seen as an adaptively variable unit roundoff.

Remark 4.3. *From our point of view, the a posteriori filtering behaves similarly as rounding in the standard IEEE 754. But it is clear that an aggressive dropping (corresponding to the large values of τ) may provide significantly different ordering represented by the matrix $\bar{P}$ than computation without dropping.*

Algorithm 5 can be then extended by the a posteriori filtering based dropping technique.

Chapter 5

Incomplete schemes: numerical experiments

In this chapter we discuss results of experiments with matrices from the real-world problems. First, we consider some small test cases and then we will also mention some experiments with large-scale matrices. It has been already shown in [1] that the *SAINV* algorithm based on the modified version of the GGS process provides the best results among the considered A-orthogonalization schemes. In order to present the full potential of the approximate inverse preconditioning based on the GGS process as well as to show its other numerical aspects (pivoting, scaling) we consider in this chapter only this algorithm. In particular, we will deal with the *incomplete* version of the generalized Gram–Schmidt process that was introduced in Chapter 4. The algorithm drops nonzero entries less (in magnitude) than an *adaptive* drop tolerance τ_i prescribed in individual major steps i of the process.

We attempt to keep the conditioning of the principal leading submatrices $\widetilde{U}_k$ as low as possible as long as possible by combining the dropping strategy with pivoting (Cholesky-based) that approximately minimizes functional (4.10). In addition, our numerical experiments employ a slight extension of the scaling by Lin and Moré [28].

As mentioned above, this chapter is composed from the two main parts. The first of them is devoted to solving small problems. These problems are studied in more detail from the point of view of the spectra of the preconditioned matrices using also some other indicators describing the preconditioned iterative method. The second part of this chapter considers larger problems in order to point out the potential of the approach based on the considered incomplete GGS process.

5.1 Definition of the test problems

Assume the system of linear algebraic equations (1.4), where as the coefficient matrix $A \in \mathbb{R}^{n\times n}$ we chose several problems from the Tim Davis collection [13].

We assume the right hand side in the form $b = Ax$, where $x = (1, \dots, 1)^T$. These linear systems are solved using the preconditioned conjugate gradient method (PCG) that uses preconditioner based on the incomplete generalized Gram–Schmidt process introduced in Chapter 4.

5.2 Stopping criterion and computational cost

Our stopping criterion for the preconditioned conjugate gradient method is based on the normalized residual (backward error)

$$\varrho^{(i)} = \frac{\|b - Ax^{(i)}\|}{\|A\|\|x^{(i)}\| + \|b\|}, \tag{5.1}$$

where $x^{(i)}$ is the ith approximation of the solution from the PCG. We consider stopping criterion $\varrho^{(i)} \leq 10^{-14}$. Note that one could use also more sophisticated stopping criterion based on estimation of the A-norm of the error (see, e.g., [34]) as well as a looser stopping tolerance. The normalized A-norm of the error is defined by

$$\varrho_A^{(i)} = \left(\frac{\langle x^{(i)} - x^*, x^{(i)} - x^* \rangle_A}{\langle x^*, x^* \rangle_A} \right)^{1/2}, \tag{5.2}$$

where x^* is the exact solution.

Main goal in developing of the preconditioning techniques for iterative methods is to achieve desired accuracy of the solution with respect to other preconditioning techniques (or in an extreme way with respect to a non-preconditioned iterative method) more efficiently. The computational time goes often hand in hand with the computational cost (measured by number floating point operations — *flop*). Although the preconditioning may lead to a faster convergence of PCG, the computational cost of iteration with preconditioning may increase. The computational cost of an iteration for the CG method (non-preconditioned) corresponds to $\mathcal{O}(\text{nnz}(A))$ and for the PCG method to $\mathcal{O}(\text{nnz}(A) + 2\,\text{nnz}(\widetilde{Z}))$. The ratio

$$\Theta = \frac{\text{nnz}(A) + 2\,\text{nnz}(\widetilde{Z})}{\text{nnz}(A)}, \tag{5.3}$$

will be called the *cost increase per iteration.* In order to show the "optimal" value of τ, we also introduce the PCG *computational cost* as

$$\Phi = \frac{(\text{nnz}(A) + 2\,\text{nnz}(\widetilde{Z}))\texttt{iters}}{\text{nnz}(A)}, \tag{5.4}$$

where `iters` denotes the iteration count to achieve desired accuracy of the solution. The quantities (5.3) and (5.4) do not always replace the total time (time to compute preconditioning and time to obtain solution using PCG) as an important indicator (on sequential computers). Both quantities (5.3) and (5.4) we have introduced here since a part of our experiments was computed in Matlab. Note that, some illustrating figures are based on the expensive *singular value decomposition.*

5.3 Small test problem

Assume the system of linear algebraic equations (1.4), where the coefficient matrix is *bcsstk07* [13]. Its sparsity pattern is depicted in Figure 5.1. It has a small dimension $n = 420$ and small number of nonzeros $\text{nnz}(A) = 7860$ that enables to show very detailed results. Nevertheless, note that we have obtained very similar results for the whole group of matrices arising from other problems of engineering.

For every problem several preconditioners $\widetilde{Z}\widetilde{Z}^T \approx A^{-1}$ using the incomplete modified GGS Algorithm are computed. We deal with variants of the modified GGS process that employ the *a posteriori filtering* as introduced in subsection 4.3.2, in more detail:

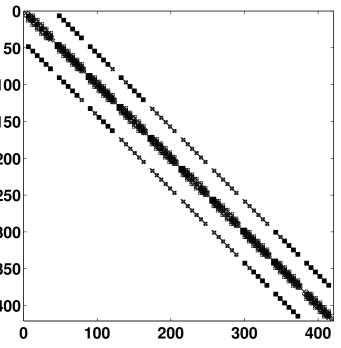


Figure 5.1: Sparsity pattern corresponding to the matrix bcsstk07.

(a) non-pivoted algorithm, original matrix,

(b) non-pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling,

(c) pivoted algorithm, original matrix,

(d) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling.

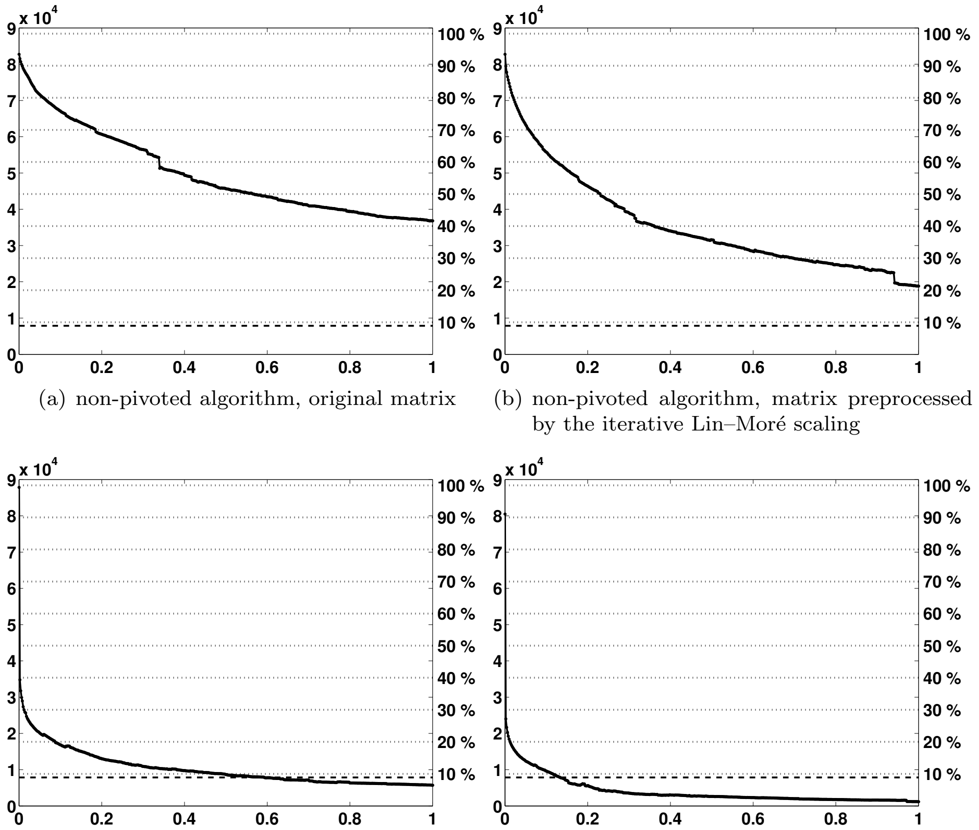
In order to see the difference among these algorithms, we depict in one figure a block of four figures as a 2×2 array where the plots corresponding to these algorithms are labeled in the same way as in intemize. Such type of description is used for the Figures 5.2, 5.3, 5.4, 5.8, 5.5, and 5.6.

Size of the test problem enables us to deliver very detailed results. We can compute preconditioners for a large number of possible accuracies τ. We assume $\tau = [0, 1]$ with an equidistant step equal to 0.002, it means approximately 500 preconditioners for every algorithm. Several figures focus only on a subset (that correspond to drop tolerances $\tau = 0.05, 0.1, 0.2, 0.4, 0.8, 1$) of the whole computation. The scaling, that is used in this section, is based on the paper by Lin and Moré [28] similarly as in Algorithm 6 by using 20 steps of the procedure. In practise, 20 steps is not realistic (too much) for a real-world computation, but this is not our interest of this section. Test problems in this section were performed using Matlab with $u \approx 1.1 \cdot 10^{-16}$.

5.3.1 Approximate inverse: effects of pivoting and scaling

From the point of view $\text{nnz}(\widetilde{Z})$ we can see from plots (a) and (b) in Figure 5.2 that the non-pivoted algorithm (even employing the iterative Lin–Moré scaling) does not provide sufficiently sparse preconditioners. Although PCG converges very fast in this case (as we can see from plots (a) and (b) in Figures 5.3 and 5.4), the preconditioning is rather close to a direct method. The quantities Θ and Φ (plots (a) and (b) in Figure 5.8) indicate very high computational cost. The reason for this fact is clear as we can see from Figure 5.5. Since we often reach $\kappa(\widetilde{U}_k) \approx \kappa(\widetilde{U}_n)$ in the first few major steps, then the a posteriori filtering allows to drop only very small entries in the later steps. Therefore, the rest of the entries has to be preserved in order to keep the chosen accuracy of the preconditioner τ.

On the other hand, we can see from plots (c) and (d) in Figure 5.2 that algorithms employing pivoting may provide rather sparse preconditioners where $\text{nnz}(\widetilde{Z}) \approx \text{nnz}(A)$. The preconditioners computed with the drop tolerance $\tau = 0.05$ and $\tau = 0.1$ imply also fast (linear) convergence (in terms of the normalized residual and the normalized A-norm of the error), see, plots (c) and (d) in Figures 5.3 and



(a) non-pivoted algorithm, original matrix

(b) non-pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling

(c) pivoted algorithm, original matrix

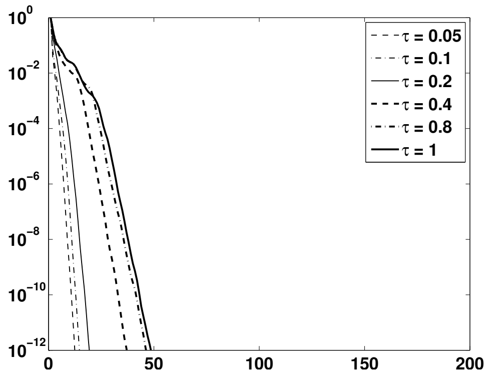
(d) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling

Figure 5.2: Sparsity of the preconditioner as a function of τ. The left axis describes $\mathrm{nnz}(\widetilde{Z})$, the right describes the ratio of fill-in with respect to the full triangular factor of the corresponding dimension. The bold dashed line denotes $\mathrm{nnz}(A)$ (in absolute value).

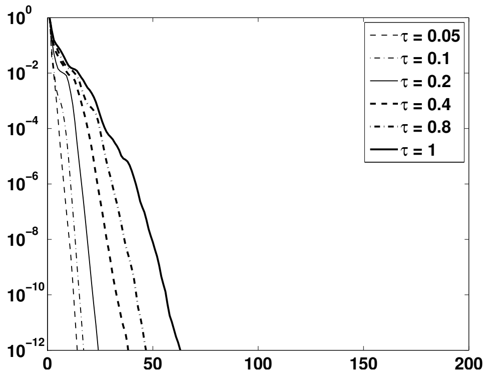
5.4. The fact that the tolerances $\tau = 0.05$ and $\tau = 0.1$ in this case lead to good preconditioners is also indicated in plots (c) and (d) in Figure 5.8. The computational cost of PCG for these values of τ attains its minimal value (if we omit cases with unreasonable high number of nonzeros). Significant benefits of pivoting can be also seen by comparing plots (c), (d) in Figure 5.5 with plots (a), (b) in Figure 5.5.

Let us deal with the eigenspectra of the leading principal submatrices of the preconditioned matrix $\widetilde{Z}^T A \widetilde{Z}$. More in detail, consider the cases that have similar PCG convergence in terms of the iteration count to achieve a desired accuracy, see Figure 5.6.

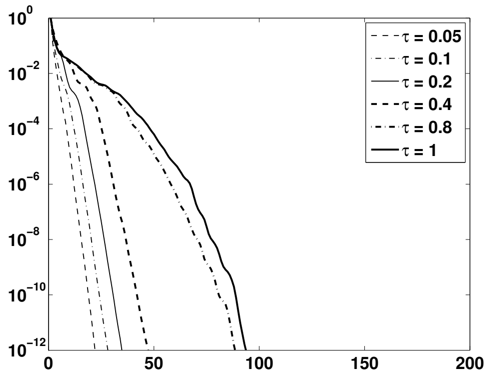
It has been shown in thesis that pivoting such that it holds (2.4) and (2.5) for the entries of $\bar{U}$ does not improve numerical properties of the computed matrices in terms of the loss of A-orthogonality (and neither in terms of the right residuals), but this fact seems to be no longer true for the incomplete decomposition as we can see from Figure 5.6. There we demonstrate the fact that the modified GGS process with the a posteriori filtering and the Cholesky-based pivoting is able to significantly improve approximation properties of $\widetilde{Z}^T A \widetilde{Z}$. It may be further improved if we employ the iterative Lin–Moré scaling. This strategies (the Cholesky-based pivoting and the iterative Lin–Moré scaling) shrinks the spectrum of the leading principal matrices of the preconditioned matrix $\widetilde{Z}^T A \widetilde{Z}$ in the subsequent steps of



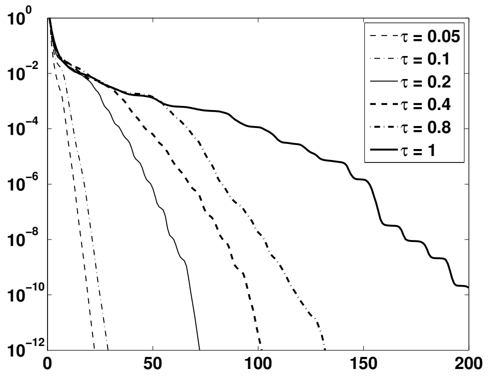
(a) non-pivoted algorithm, original matrix



(b) non-pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling



(c) pivoted algorithm, original matrix



(d) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling

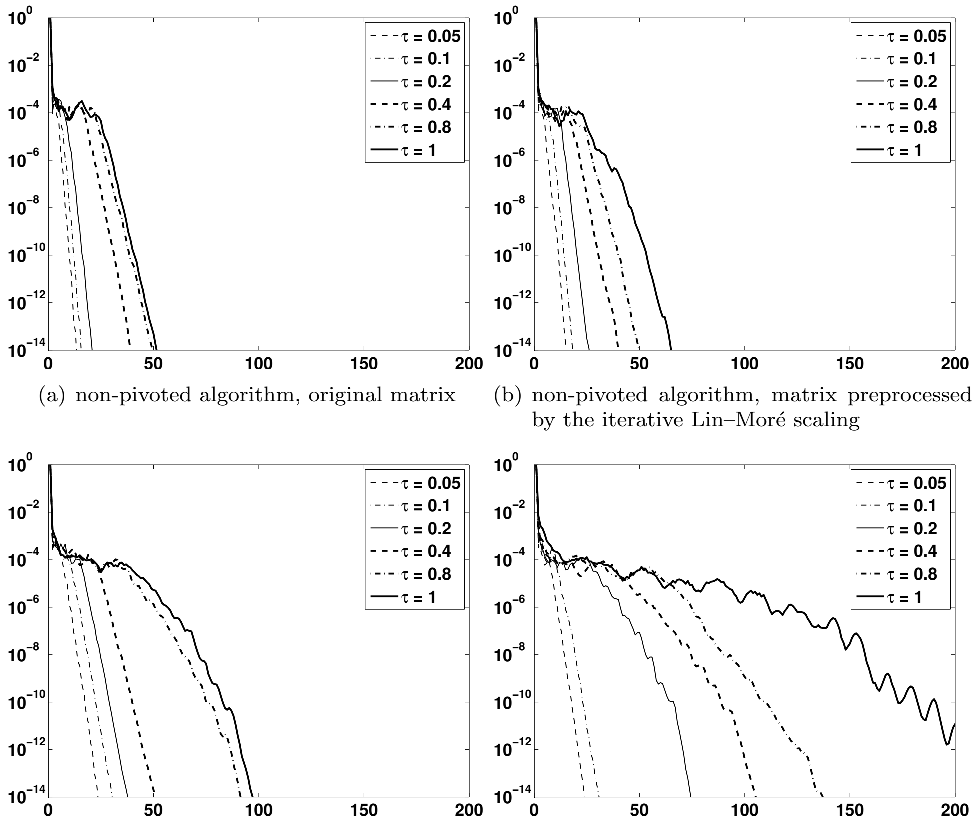
Figure 5.3: Convergence of PCG in terms of the normalized A-norm of the error as a function of iteration number for several values of τ.

the decomposition much more. Employing pivoting strategies and scaling, we are able to represent the approximate inverse with a significantly smaller number of nonzeros. It can be also seen that the loss of A-orthogonality that is for all the depicted cases approximately equal to 0.7. Namely, the A-orthogonality among the column vectors in $\widetilde{Z}$ is well preserved.

Incomplete modified GGS with Cholesky-based pivoting combined with Lin–Moré scaling

Let us consider plot (d) in Figures 5.3 and 5.4. In both cases, the significant gap in convergence corresponds to $\tau = 0.1$ and $\tau = 0.2$ (the gap can also be seen from computational cost in the plot (d) in Figure 5.8). The eigenspectra of the leading principal submatrices of the preconditioned matrix $\widetilde{Z}^T A \widetilde{Z}$ for these two cases can be found in Figure 5.7. Between these two plots we can see the significant differences, the A-orthogonality for the plot (b) is completely lost ($\|\widetilde{Z}^T A \widetilde{Z} - I\| > 1$).

From Figure 5.8 we can see that the cost increase per iteration seems to be "continuous" and nearly monotonically decreasing function. This also indicates the "continuity" of the sparsity patterns of $\widetilde{Z}$. On the other hand, a significant gap can be observed in the computational cost. Let us discuss the computational cost curve in plot (d) in Figure 5.8 in more detail. The first part (related approximately to $\tau \in [0, 0.02]$) corresponds to a very accurate computation of the approximate inverse. The lack of sparsity of the preconditioner that corresponds to the first part of the



(a) non-pivoted algorithm, original matrix

(b) non-pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling

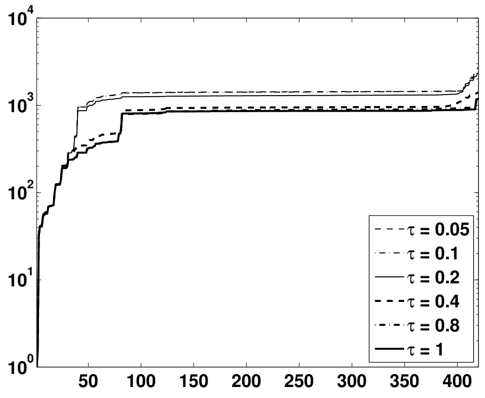
(c) pivoted algorithm, original matrix

(d) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling

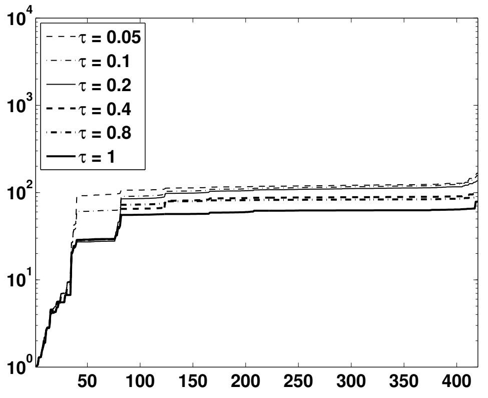
Figure 5.4: Convergence of PCG in terms of the normalized residuals as a function of iteration number for several values of τ.

curve in plot (d) in Figure 5.8, makes it hard to usable in practice. Its second part (related approximately to $\tau \in [0.02, 0.15]$) corresponds to sparse preconditioner that leads to nearly linear convergence of PCG. Preconditioners that correspond to the second part of this curve are the optimal ones. The third part of the curve (related approximately to $\tau \in [0.15, 1]$) corresponds to the case where the orthogonality is completely lost ($\|\widetilde{Z}^T A \widetilde{Z} - I\| > 1$). The preconditioning is robust also in this case, but it does not lead qualitatively to the similar convergence as in previous part of the curve. The fourth and also the last part of the curve (related approximately to $\tau \in [0.95, 1]$) corresponds to nearly diagonal preconditioners (Jacobi-like), however, computation of such preconditioners using GGS is not efficient.

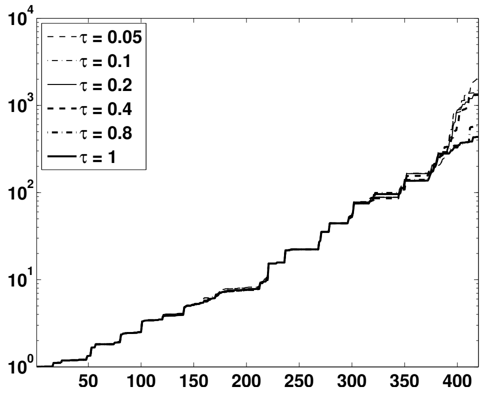
Let us also deal with the sparsity patterns of the matrix $\widetilde{P}^T \widetilde{Z}$ (multiplying from the left by $\widetilde{P}^T$ is performed in order to have the matrix in the upper triangular form). Figure 5.9 is formed from 6 plots (as an 3×2 array) corresponding to sparsity patterns of the matrices $\widetilde{P}^T \widetilde{Z}$ and computed by the incomplete modified GGS algorithm as specified in item (d). We can see the following intuitive fact (that also partially arises from the bordering scheme presented in Chapter 3). For small and well conditioned (the interlacing theorem for singular values) leading principal submatrices A_k (or $[\widetilde{P}^T A \widetilde{P}]_k$) is not necessary to use a full potential of double precision arithmetic – major the part of the operations in finite precision arithmetic can be truncated to zero and the diagonal preconditioner is sufficiently good. But with a growing size of the leading principal submatrices A_k (or $[\widetilde{P}^T A \widetilde{P}]_k$)



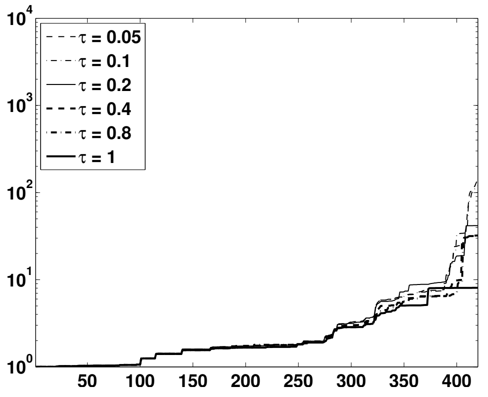
(a) non-pivoted algorithm, original matrix



(b) non-pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling



(c) pivoted algorithm, original matrix

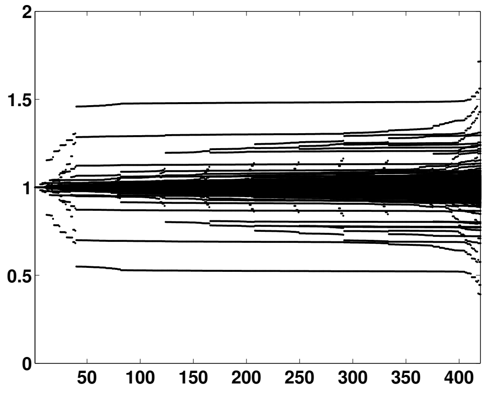


(d) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling

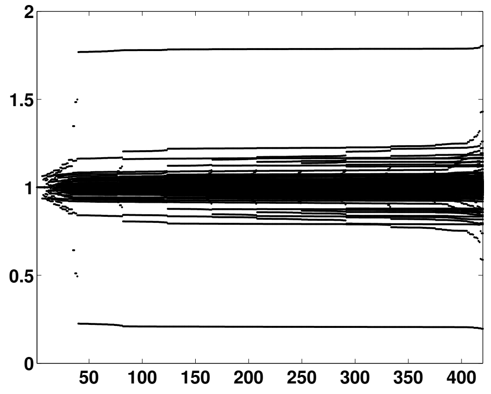
Figure 5.5: Local condition numbers $\kappa(\widetilde{U}_k)$ as a function of k for several values of τ.

it is necessary to increase the local precision τ_i that leads to off-diagonal fill-in in $\widetilde{P}^T\widetilde{Z}$.

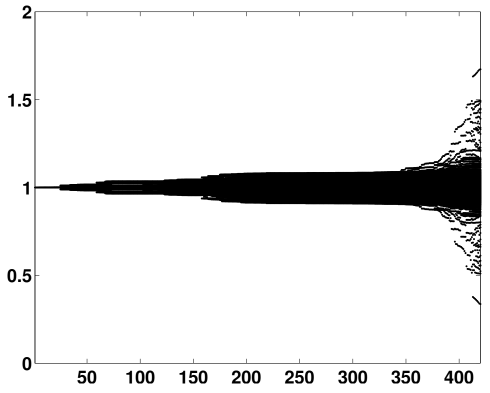
Remark 5.1. *The most aggressive dropping in the first few steps leads to the diagonal preconditioning that is equivalent to the Jacobi preconditioner. Therefore, also $\tau = \infty$ reduces whole incomplete GGS preconditioner to the Jacobi (diagonal) preconditioner.*



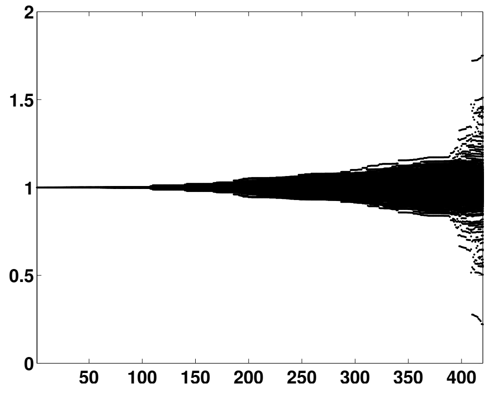
(a) non-pivoted algorithm, original matrix, $\tau = 0.2$, $\mathrm{nnz}(\widetilde{Z}) = 60659$, and `iters` $= 20$



(b) non-pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling, $\tau = 0.1$, $\mathrm{nnz}(\widetilde{Z}) = 55773$, and `iters` $= 18$

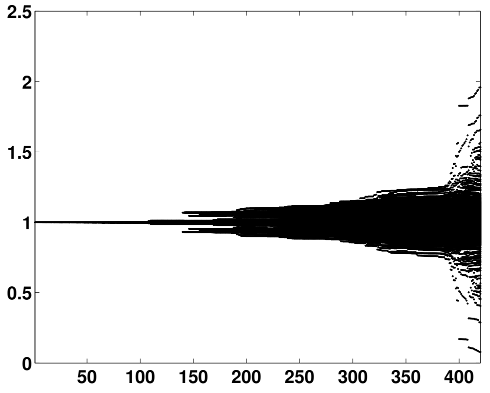


(c) pivoted algorithm, original matrix, $\tau = 0.05$, $\mathrm{nnz}(\widetilde{Z}) = 20125$, and `iters` $= 23$

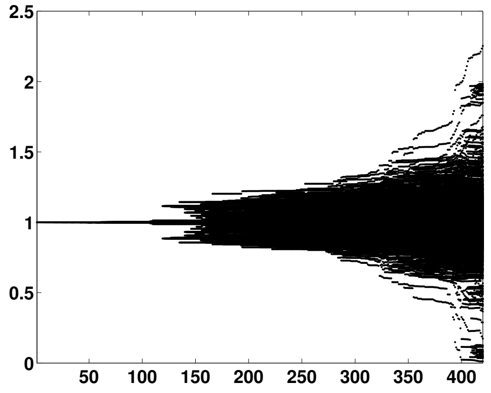


(d) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling, $\tau = 0.05$, $\mathrm{nnz}(\widetilde{Z}) = 12393$, and `iters` $= 23$

Figure 5.6: Eigenvalues of the leading principal submatrices of the matrix $\widetilde{Z}^T A \widetilde{Z}$. These plots correspond to different tolerances τ.

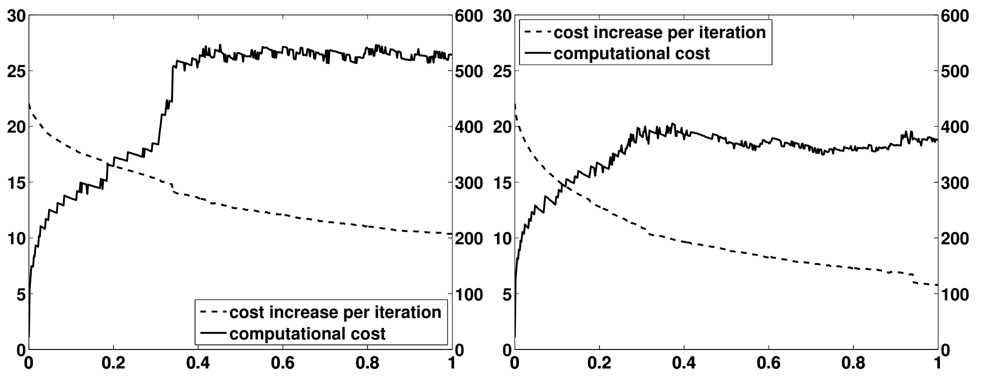


(a) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling, $\tau = 0.1$, $\mathrm{nnz}(\widetilde{Z}) = 9663$, and `iters` $= 30$



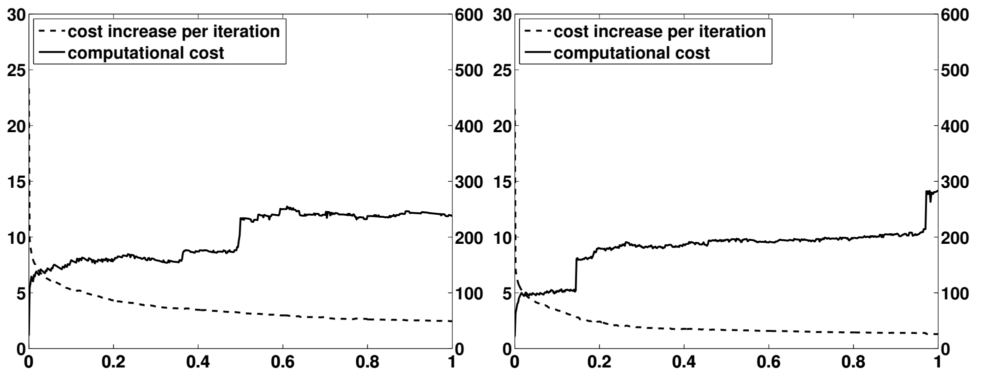
(b) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling, $\tau = 0.2$, $\mathrm{nnz}(\widetilde{Z}) = 5624$, and `iters` $= 74$

Figure 5.7: Eigenvalues of the leading principal submatrices of the matrix $\widetilde{Z}^T A \widetilde{Z}$.



(a) non-pivoted algorithm, original matrix

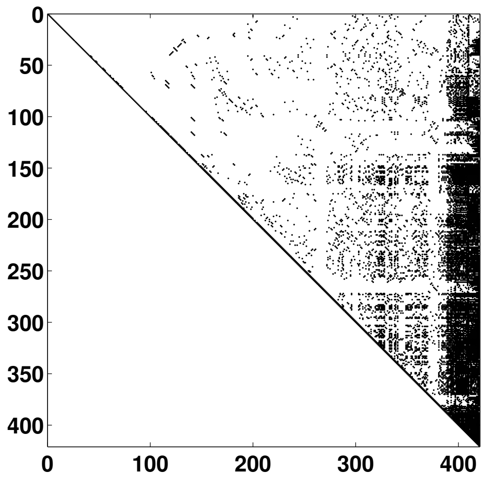
(b) non-pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling



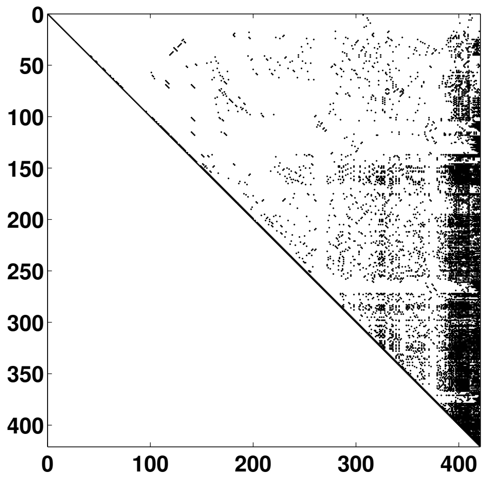
(c) pivoted algorithm, original matrix

(d) pivoted algorithm, matrix preprocessed by the iterative Lin–Moré scaling

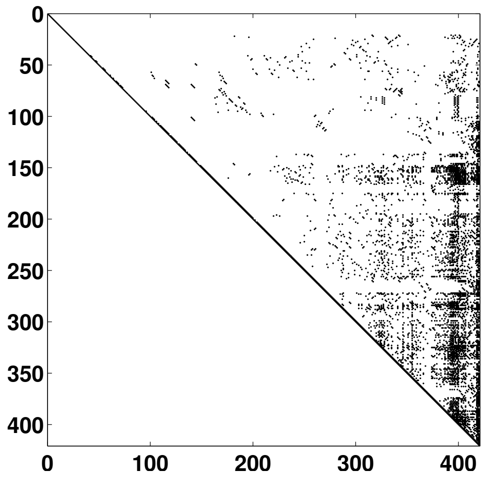
Figure 5.8: Cost increase per iteration (5.3) as a function of τ *dashed line* and the computational cost of PCG (5.4) as a function of τ *solid line*.



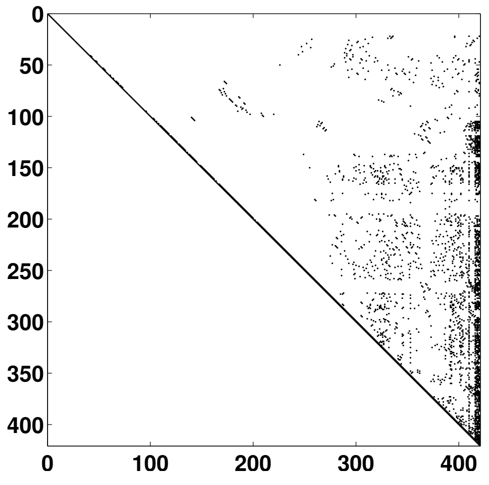
(a) $\tau = 0.05$



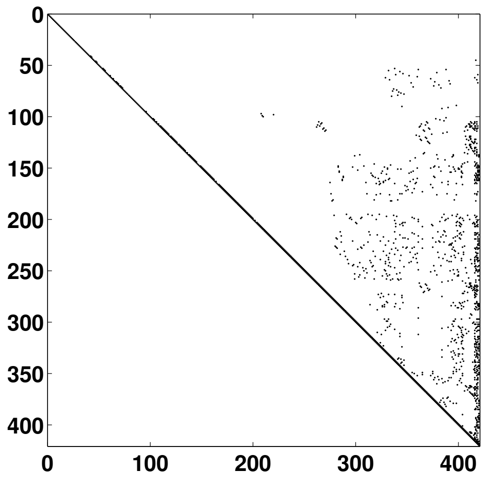
(b) $\tau = 0.1$



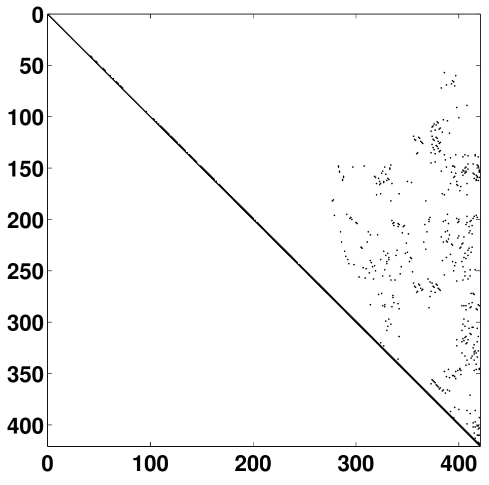
(c) $\tau = 0.2$



(d) $\tau = 0.4$



(e) $\tau = 0.8$



(f) $\tau = 1$

Figure 5.9: Sparsity patterns of the matrices $P^T \widetilde{Z}$ for several variants of τ that are computed by the modified GGS algorithm employing pivoting, matrix preprocessed by the iterative Lin–Moré scaling (as specified in (d)).

5.4 Large test problems

In this section we consider larger problems described in Table 5.4. Note that, incomplete GGS Algorithm (considered also in this section) can be implemented efficiently in the fully sparse mode [2]. Also $\kappa(\widetilde{U}_k)$ can be estimated efficiently. In order to deploy a fast code, we use the optimistic estimation $\kappa(\widetilde{U}_k) \approx \kappa_S(\widetilde{U}_k)$. Note that, in these experiments we do not use scaling. Experiments in this section are performed using the Fortran code [35].

Table 5.1: Test matrices from Tim Davis collection [13]

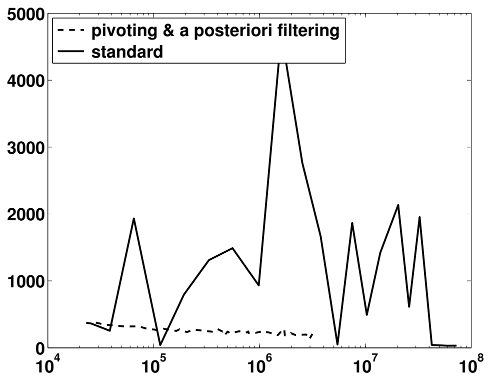
Matrix	Dimension $[n]$	nnz(A)	Description
bcsstk36	23 052	1 143 140	Stiffness matrix, automobile shock absorber assembly
ldoor	952 203	42 493 817	Structural 2D/3D problem
af_shell8	504 855	17 579 155	Olaf Schenk, sheet metal forming
nasasrb	54 870	2 677 324	Structure from NASA Langley, shuttle rocket booster
oilpan	73 752	2 148 558	Test matrix from Inpro

In the previous section, we have shown how the Cholesky-based pivoting and the iterative Lin–Moré scaling may improve quality of the approximate inverse preconditioning that uses dropping strategy based on the a posteriori filtering developed in subsection 4.3.2.

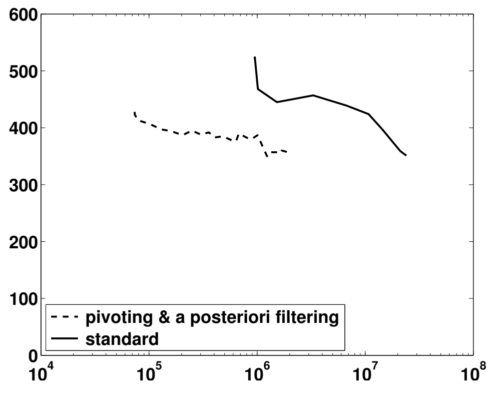
Up to now, dropping techniques in the incomplete GGS process in the published papers have been based on absolute or relative dropping similarly to incomplete Cholesky factorization (IC(τ)). The a posteriori filtering is more related to relative dropping called here the *standard* incomplete decomposition ($\tau_k = \tau$). Therefore, we compare these two approaches to compute the approximate inverse preconditioning, see, Figure 5.10.

The results (depicted in Figure 5.10) with the new approach seem to be good from more possible points of view: iteration count, nnz$(\widetilde{Z})$. Based on the results for these test matrices from structural engineering, we see that this approach leads to rather powerful and still sparse preconditioners. In some cases as depicted in plot (e) in Figure 5.10, the new approach may deliver slightly worse results. But note that worse results may be also due to inaccurate estimation of $\kappa(\widetilde{U}_k)$ that is in this case estimated in the simplest way as $\kappa(\widetilde{U}_k) \approx \kappa_S(\widetilde{U}_k)$. More sophisticated condition estimation may further improve properties of the approximate inverse.

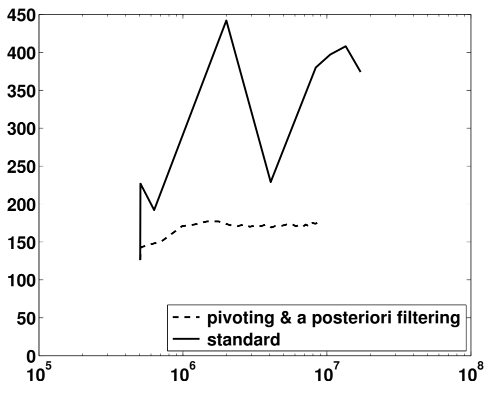
In general, we can see that the new approach employing the Cholesky-based pivoting with the a posteriori filtering may enhance standard dropping rules [3]. Although in a lot of cases we can observe faster convergence for a given sparsity of the preconditioner nnz$(\widetilde{Z})$ with respect to standard approaches, the more significant property of this approach seems to be that the preconditioner is nearly always much less sensitive to the input τ. Therefore, finding the proper value τ is very often much easier.



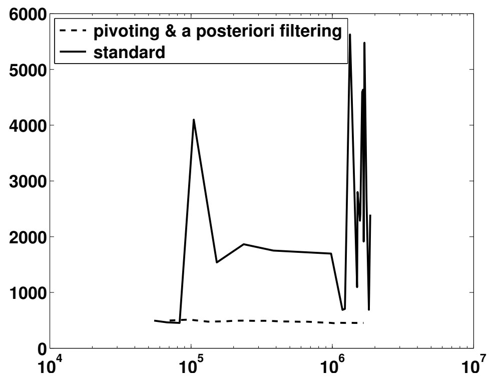
(a) matrix *bcsstk36*



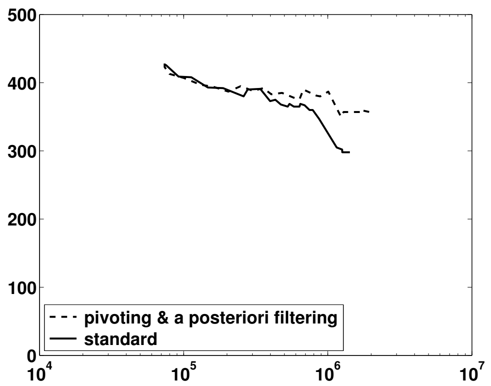
(b) matrix *ldoor*



(c) matrix *af_shell8*



(d) matrix *nasasrb*



(e) matrix *oilpan*

Figure 5.10: Iteration count (`iters`) as a function of the sparsity of the preconditioner $\mathrm{nnz}(\widetilde{Z})$ for the standard incomplete decomposition ($\tau_k = \tau$) and for the decomposition employing the Cholesky-based pivoting and the a posteriori filtering ($\tau_k = \tau/\kappa(\widetilde{U}_k)$).

Chapter 6

Conclusions and open questions

Here we summarize results presented in the thesis. We formulate some open questions and mention some possible directions for further work.

6.1 Conclusions

We give a new insight into the behavior of the generalized Gram–Schmidt based preconditioning. In more detail, the new approach deals with dropping strategies for a class of AINV-type incomplete decompositions [2] that are based on the generalized Gram–Schmidt process. Its behavior in finite precision arithmetic has been discussed in [31]. This analysis enables better understanding of the incomplete process. Based on the results in [31] and also some new error bounds developed here, we propose a new dropping strategy to construct the incomplete decomposition.

We develop the GGS algorithm in the left-looking form that employ Cholesky-based pivoting which does not significantly increase its computational cost.

We show that all the variants of the GGS process in finite precision arithmetic (without iterative refinement) compute the matrix $\bar{Z}$ using the numerically equivalent way that corresponds to left-looking (or numerically equivalent right-looking) scheme for computing inverse $X = \bar{Z}$ of the upper triangular matrix $\bar{U}$ from the equation $X\bar{U} = I$. Based on the theory and also on the numerical experiments, we demonstrate (for the *complete* GGS algorithm) that the indicators that correspond to the stability of the preconditioner (loss of A-orthogonality $\|\bar{Z}^T A\bar{Z} - I\|$ or $\|U\bar{Z} - I\|$ that is also related to right residuals $\|\bar{U}\bar{Z} - I\|$) are hard to improve using pivoting such that it holds (2.4) and (2.5) for the entries of $\bar{U}$. Although the component-wise error bounds of the right residuals are different for pivoted and non-pivoted algorithms, the associated spectral norm of the bounds exhibit essentially the same behavior.

Nevertheless, the component-wise error bounds for GGS with $Z^{(0)} = I$ that are based on the bordering scheme of the upper triangular matrices bring a new insight into the GGS process. This motivate us to introduce a new strategy (called here the *a posteriori filtering*) that naturally puts adaptivity into the dropping. The a posteriori filtering is based on the idea to discard all the entries that increase the error in decomposition up to desired accuracy. Our implementation is such that this discarding of the entries behaves similarly as the rounding error in finite precision arithmetic. Therefore, the new dropping strategy can be seen as computation with a variable machine precision that is controlled by global quantities (extremal singular values) of the leading principal submatrices $\widetilde{U}_k$. These quantities play also (based partly on the theory and on the observations) a significant role for the errors in the *complete* GGS process. In this way the constructed dropping strategy (the a posteriori filtering) seems to be a link between *complete* algorithms in finite precision arithmetic and *incomplete* algorithms.

Although the pivoting does not seem to be beneficial for improving the loss of A-orthogonality or the decrease right residuals for the *complete* GGS process, this is no longer true for the *incomplete* GGS process. Employing pivoting makes the *incomplete* GGS process more stable and in addition it may approximate the factor of the inverse triangular factorizations of A^{-1} with less number of nonzeros by preserving the magnitude of $\|\widetilde{Z}^T A \widetilde{Z} - I\|$. In addition, if we combine pivoting and the iterative Lin–Moré scaling, sparsity of $\widetilde{Z}$ may be improved even more.

The theoretical insight is accompanied by some experimental results for matrices arising from structural mechanics. The results point out that the theoretically motivated a posteriori filtering with the Cholesky-based pivoting and the iterative Lin–Moré scaling leads to a rather powerful approximate inverse preconditioning. In particular, it seems that the new dropping strategy provides preconditioners that are sparse and powerful at the same time. Let us summarize the key features of our preconditioning:

1. It is based on a specific combination of the modified GGS algorithm in the left-looking form and the Cholesky-based pivoting;
2. It is theoretically motivated;
3. It works adaptively;
4. The dropping behaves similarly as rounding in finite precision arithmetic (variable machine precision);
5. It is nearly always much less sensitive to input with respect to standard approaches;
6. It is more easier search for the dropping parameter (τ) with respect to standard approaches.

This thesis represents a continuation of our research published in [26].

6.2 An open questions and possible directions for further research

Although our component-wise error bounds for the right residuals seem to be tight (if we take the whole bound into the spectral norm), for a general matrix we are not able to show its equivalence with $\mathcal{O}(n)u\kappa^{1/2}(A)$. This problem is also related to the theoretical error bound for the loss of A-orthogonality that seems to be $\approx \mathcal{O}(n)u\kappa(A)$.

In Section 5.4 we have dealt only with the simplest estimation of $\kappa(\widetilde{U}_k)$ based on $\kappa_S(\widetilde{U}_k)$. Open question is how significantly can be the incomplete scheme improved using some sophisticated incremental condition estimator.

The further direction of the research may be to generalize our dropping strategy (the a posteriori filtering) to another preconditioning, e.g., the incomplete Cholesky factorization (IC).

6.3 List of publications and conference talks of the author of this thesis related to the subject

Papers in international journals

[J2] M. Rozložník, A. Smoktunowicz, J. Kopal: *A note on iterative refinement for seminormal equations*, Applied Numerical Mathematics, Volume 75, pp. 167–174, 2014.

[J1] M. Rozložník, M. Tůma, A. Smoktunowicz, J. Kopal: *Numerical stability of orthogonalization methods with a non-standard inner product*, BIT Numerical Mathematics, 52 (4), pp. 1035–1058, 2012.

Papers submitted to international journals

[J3] J. Kopal, M. Rozložník, M. Tůma: *Approximate inverse preconditioners with adaptive dropping*, submitted to Advances in Engineering Software, 2013.

Proceedings contributions

[P8] J. Kopal, M. Rozložník, M. Tůma: *Approximate Inverse Preconditioning for the Conjugate Gradient Method*, in B.H.V. Topping, P. Iványi (Editors), Proceedings of the Third International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering. Civil-Comp Press, Stirlingshire, United Kingdom, 2013, paper 21.

Awarded the Young Researcher Best Paper Prize in the category: mathematics or computer science.

[P7] J. Kopal, M. Rozložník, A. Smoktunowicz: *On iterative refinement for seminormal equations,* SNA '12, Seminar on Numerical Analysis, Liberec, TUL (2012), pp. 97–99.

[P6] J. Kopal, M. Rozložník, M. Tůma: *Orthogonalization with a non-standard inner produt and approximate inverse preonditioning*, In: Seminar on Numerical Analysis '11 (R. Blaheta, J. Starý, Eds.), Rožnov pod Radhoštěm, Institute of Geonics, AS CR (2011), pp. 61–64.

[P5] M. Hokr, J. Kopal, J. Březina, P. Rálek: *Sensitivity of Results of the Water Flow Problem in a Discrete Fracture Network with Large Coefficient Differences*, Numerical Methods And Applications, Lecture Notes in Computer Science, Vol 6046 (2011), pp. 420–427.

[P4] J. Kopal: *Analysis of Algebraic Preconditioning Based on Different Variants of the Gram–Schmidt Algorithm*, (part II, in Czech), Proc. of XVI. PhD. Conference '11 (F. Hakl, Ed.), Praha, ICS, AS CR & Matfyzpress (2011), pp. 73–79.

[P3] J. Kopal: *Analysis of Algebraic Preconditioning Based on Different Variants of the Gram–Schmidt Algorithm* (part I, in Czech), Proc. of XV. PhD. Conference '10 (F. Hakl, Ed.), Praha, ICS, AS CR & Matfyzpress (2010), pp. 72–77.

[P2] M. Hokr, J. Kopal, J. Havlíček: *Řešení úlohy proudní v rozsáhlé diskrétní síti puklin v kontextu sdružených úloh proudění-mechanika*, SNA '09, Seminar of Numerical Analysis 09 (R. Blaheta, J. Starý, Eds.), Modelling an Simulation of Challenging Engineering Problems. Ostrava, Institute of Geonics AS CR, 2009. pp. 46–49.

[P1] J. Havlíček, M. Hokr and J. Kopal: *Non-linear adsorption in a multidimensional and double-porosity model of fractured rock solute transport* Predictions for Hydrology, Ecology, and Water Resources Management: Using Data and Models to Benefit Society (Bruthans, Kovář, and Hrkal, eds.), Czech Assoc. Hydrogeol., 2008, pp. 167–170.

International conferences (talks and posters)

[I4] J. Kopal, M. Rozložník, M. Tůma: *Approximate inverse preconditioning with adaptive dropping*, Sparse Days Meeting 2013 (Sparse Days 2013), Toulouse, France, June 17–18, 2013.

[I3] J. Kopal, M. Rozložník, M. Tůma: *Approximate inverse preconditioning for the conjugate gradient method*, High Performance Computing in Science and Engineering (HPCSE 2013), Hotel Soláň, Czech Republic, May 27–30, 2013.

[I2] J. Kopal, M. Rozložník, M. Tůma: *Approximate Inverse Preconditioning for the Conjugate Gradient Method*, The Third International Conference on Parallel, Distributed, Grid and Grid Computing for Engineering (PARENG2013), Pécs, Hungary, March 25–27, 2013.

[I1] J. Kopal, M. Rozložník, M. Tůma: *Generalized Gram–Schmidt-based approximate inverse preconditioning for the conjugate gradient method*, XII GAMM Workshop on Applied and Numerical Linear Algebra, Liblice, Czech Republic, September 2–5, 2012.

Local conferences (talks and posters)

[L5] J. Kopal, M. Rozložník, A. Smoktunowicz: *On iterative refinement for seminormal equations,* SNA '12, Seminar on Numerical Analysis, Liberec, January 23–27, 2012.

[L4] J. Kopal, M. Rozložník, M. Tůma: *Orthogonalization with a non-standard inner produt and approximate inverse preonditioning*, SNA '11, Seminar on Numerical Analysis, Rožnov pod Radhoštěm, January 24–28, 2011.

[L3] J. Kopal: *Analysis of Algebraic Preconditioning Based on Different Variants of the Gram–Schmidt Algorithm* (in Czech), Proc. of XVI. PhD. Conference, Jizerka, October 5–7, 2010.

[L2] J. Kopal: *Analysis of Algebraic Preconditioning Based on Different Variants of the Gram–Schmidt Algorithm* (in Czech), Proc. of XV. PhD. Conference, Hemanice v Podještědí, September 29 – October 1, 2010.

[L1] M. Hokr, J. Kopal, J. Havlíček: *Řešení úlohy proudění v rozsáhl diskrétní síti puklin v kontextu sdružených úloh proudění-mechanika*, SNA '09, Seminar on Numerical Analysis, Ostrava, February 2–6, 2009.

Seminar lectures

[S8] J. Kopal: *Approximate inverse preconditioning with adaptive dropping*, Seminar at Department of Mathematics, FP, TUL, March 10, 2014.

[S7] J. Kopal: *Approximate inverse preconditioning with adaptive dropping*, Seminar at Institute of Novel Technologies and Applied Informatics, FM, TUL, November 27, 2013.

[S6] J. Kopal: *Generalized Gram–Schmidt-based approximate inverse preconditioning for the conjugate gradient method*, Seminar at Department of Mathematics, FP, TUL, October 15, 2012.

[S5] J. Kopal: *Generalized Gram–Schmidt-based approximate inverse preconditioning for the conjugate gradient method*, Seminar at Institute of Novel Technologies and Applied Informatics, FM, TUL, October 3, 2012.

[S4] J. Kopal: *Selected numerical aspects of algebraic preconditioning computed by generalized Gram–Schmidt process*, Seminar at Institute of Novel Technologies and Applied Informatics, FM, TUL, July 27, 2012.

[S3] J. Kopal: *Analysis of algebraic preconditioning based on different variants of the Gram–Schmidt algorithm III*, Seminar at Institute of Novel Technologies and Applied Informatics, FM, TUL, April 13, 2011.

[S2] J. Kopal: *Analysis of algebraic preconditioning based on different variants of the Gram–Schmidt algorithm II*, Seminar at Institute of Novel Technologies and Applied Informatics, FM, TUL, October 6, 2010.

[S1] J. Kopal: *Analysis of algebraic preconditioning based on different variants of the Gram–Schmidt algorithm I*, Seminar at Institute of Novel Technologies and Applied Informatics, FM, TUL, May 19, 2010.

Bibliography

[1] M. Benzi, J. K. Cullum, and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM J. Sci. Comput.*, 22(4):1318–1332, 2000.

[2] 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.

[3] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math.*, 30(2-3):305–340, 1999.

[4] M. Benzi and M. Tůma. A robust incomplete factorization preconditioner for positive definite matrices. *Numer. Linear Algebra Appl.*, 10(5-6):385–400, 2003.

[5] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 27(4):930–951, 2006.

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

[7] C. H. Bischof and P. T. P. Tang. Robust incremental condition estimation. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 1991.

[8] C. H. Bischof and P. T. P. Tang. Generalizing incremental condition estimation. *Numer. Linear Algebra Appl.*, 1(2):149–163, 1992.

[9] Å. Björck. Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT*, 7:1–21, 1967.

[10] E. G. Boman and B. Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 25(3):694–717, 2003.

[11] S. Chandrasekaran and I. C. F. Ipsen. On the sensitivity of solution components in linear systems of equations. *SIAM J. Matrix Anal. Appl.*, 16(1):93–112, 1995.

[12] E. Chow and Y. Saad. Experimental study of ILU preconditioners for indefinite matrices. *J. Comput. Appl. Math.*, 86(2):387–414, 1997.

[13] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1–25, 2011. avaiable online at http://www.cise.ufl.edu/research/sparse/matrices.

[14] J. Duintjer Tebbens and M. Tůma. On incremental condition estimators in the 2-norm. *SIAM J. Matrix Anal. Appl.*, 35:174–197, 2014.

[15] T. Dupont, R. P. Kendall, and H. H. Jr. Rachford. An approximate factorization procedure for the solving self-adjoint elliptic difference equations. *SIAM J. Numer. Anal.*, 5:559–573, 1968.

[16] L. Fox, H. D. Huskey, and J. H. Wilkinson. Notes on the solution of algebraic linear simultaneous equations. *Quart. J. Mech. and Appl. Math.*, 1:149–173, 1948.

[17] L. Giraud, J. Langou, and M. Rozložník. The loss of orthogonality in the Gram-Schmidt orthogonalization process. *Comput. Math. Appl.*, 50(7):1069–1075, 2005.

[18] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore and London, third ed. edition, 1996.

[19] M. Gulliksson. Backward error analysis for the constrained and weighted linear least squares problem when using the weighted QR factorization. *SIAM J. Matrix Anal. Appl.*, 16(2):675–687, 1995.

[20] M. Gulliksson. On the modified Gram-Schmidt algorithm for weighted and constrained linear least squares problems. *BIT*, 35(4):453–468, 1995.

[21] M. Gulliksson and P.-Å. Wedin. Modifying the QR-decomposition to constrained and weighted linear least squares. *SIAM J. Matrix Anal. Appl.*, 13(4):1298–1313, 1992.

[22] I. Gustafsson. A class of first order factorization methods. *BIT*, 18(2):142–156, 1978.

[23] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–435, 1952.

[24] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

[25] N. Knight. Triangular Matrix Inversion: a survey of sequential approaches. Technical report, University of California, Berkeley, http://www.cs.berkeley.edu/∼knight/knight_math221_paper.pdf, 2009.

[26] J. Kopal, M. Rozložník, and M. Tůma. Approximate inverse preconditioning for the conjugate gradient method. In B. H. V. Topping and P. Iványi, editors, *Proceedings of the Third International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, Stirlingshire, United Kingdom, 2013. Civil-Comp Press. paper 21.

[27] N. Li and Y. Saad. Crout versions of ILU factorization with pivoting for sparse symmetric matrices. *Electron. Trans. Numer. Anal.*, 20:75–85, 2005.

[28] C.-J. Lin and J. J. Moré. Incomplete Cholesky factorizations with limited memory. *SIAM J. Sci. Comput.*, 21(1):24–45, 1999.

[29] B. R. Lowery and J. Langou. Stability analysis of QR factorization in an oblique inner product. *submited*, 2014.

[30] M. Rozložník, F. Okulicka-Dluzewska, and A. Smoktunowicz. Indefinite orthogonalization with rounding errors. *submitted*, 2013.

[31] M. Rozložník, M. Tůma, A. Smoktunowicz, and J. Kopal. Rounding error analysis of orthogonalization with a non-standard inner product. *BIT Numer Math*, 52:1035–1058, 2012.

[32] R. D. Skeel. Scaling for numerical stability in Gaussian elimination. *J. Assoc. Comput. Mach.*, 26(3):494–526, 1979.

[33] A. Smoktunowicz, J. L. Barlow, and J. Langou. A note on the error analysis of classical Gram-Schmidt. *Numer. Math.*, 105(2):299–313, 2006.

[34] Z. Strakoš and P. Tichý. Error estimation in preconditioned conjugate gradients. *BIT*, 45:789–817, 2005.

[35] M. Tůma. Sparslab software. *Academy of Sciences of the Czech Republic, Institute of Computer Science*, 2014. avaiable online at http://www2.cs.cas.cz/∼tuma/.

[36] S. J. Thomas. A block algorithm for orthogonalization in elliptic norms. *Lecture Notes in Computer Science*, 634:379–385, 1992.

[37] S. J. Thomas and R. V. M. Zahar. Efficient orthogonalization in the m-norm. *Congressus Numerantium*, 80:23–32, 1991.

[38] S. J. Thomas and R. V. M. Zahar. An analysis of orthogonalization in elliptic norms. *Congressus Numerantium*, 86:193–222, 1992.

[39] R.C. Thompson. Principal submatrices IX: Interlacing inequalities for singular values of submatrices. *Linear Algebra Appl.*, 5(1):1–12, 1972.

[40] L. N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators.* Princeton University Press, 2005.

[41] various. *HSL(2013). A collection of Fortran codes for large scale scientific computation.*