DOI: 10.2478/auom-2020-0024
An. Şt. Univ. Ovidius Constanţa

sciendo
Vol. 28(2),2020, 149–159

A note on adaptivity in factorized approximate inverse preconditioning*

Jiří Kopal, Miroslav Rozložník, and Miroslav Tůma

Abstract

The problem of solving large-scale systems of linear algebraic equations arises in a wide range of applications. In many cases the preconditioned iterative method is a method of choice. This paper deals with the approximate inverse preconditioning AINV/SAINV based on the incomplete generalized Gram–Schmidt process. This type of the approximate inverse preconditioning has been repeatedly used for matrix diagonalization in computation of electronic structures but approximating inverses is of an interest in parallel computations in general. Our approach uses adaptive dropping of the matrix entries with the control based on the computed intermediate quantities. Strategy has been introduced as a way to solve difficult application problems and it is motivated by recent theoretical results on the loss of orthogonality in the generalized Gram–Schmidt process. Nevertheless, there are more aspects of the approach that need to be better understood. The diagonal pivoting based on a rough estimation of condition numbers of leading principal submatrices can sometimes provide inefficient preconditioners. This short study

Key Words: sparse approximate inverse preconditioners, approximate factorization, generalized Gram-Schmidt process.

2010 Mathematics Subject Classification: 15A23, 65E50, 65F05, 65F25.

This paper has been presented at the 12th Workshop on Mathematical Modelling of Environmental and Life Sciences Problems, Constanta, October 24-28, 2018 . It was accepted for publication as a part of the Proceedings by a peer-review procedure coordinated by the editors Stelian Ion, Elena Pelican and Constantin Popa. Sponsored by BITDEFENDER Bucharest.

Received: 10.07.2019.

Accepted: 16.12.2019.

*The second author was partially supported by the Czech Science Foundation project GA17-12925S, the third author was partially supported by the Czech Science Foundation project GA18-12719S.

proposes another type of pivoting, namely the pivoting that exploits incremental condition estimation based on monitoring both direct and inverse factors of the approximate factorization. Such pivoting remains rather cheap and it can provide in many cases more reliable preconditioner. Numerical examples from real-world problems, small enough to enable a full analysis, are used to illustrate the potential gains of the new approach.

1 Introduction

Solution of many problems in science and engineering reduces to solving systems of linear algebraic equations

$$Ax = b,\ A \in \mathbb{R}^{n\times n},\ x \in \mathbb{R}^n, b \in \mathbb{R}^n. \tag{1}$$

Here, A is the system matrix, x is the vector of unknowns and b is the right-hand-side vector. We assume that the system matrix A is symmetric and positive definite (SPD). Problems that arise from discretization of partial differential equations usually provide structured systems of linear algebraic equations that are sparse, but they can be also fully unstructured.

Direct methods, in particular the sparse Cholesky factorization, represent standard methods of choice. The Cholesky factorization of the matrix can be written as $A = U^T U$, where U is an upper triangular matrix with positive diagonal entries. Another class of solution methods is represented by iterative Krylov subspace methods, e.g., in SPD case the conjugate gradient method, that can be in many cases very efficient. Nevertheless, in order to increase the potential of the Krylov subspace methods, a good preconditioner is a must. There are two main classes of useful preconditioners that are called as direct and inverse preconditioners. An incomplete variant of the Cholesky factorization $A \approx \tilde{U}^T \tilde{U}$ may be considered as a standard representative of the class of direct preconditioners. In this case forward and backward substitution steps have to be performed in every iteration step. This represents a bottleneck in a parallel computational environment. Such a drawback can be avoided by using an inverse preconditioner. Its construction is, in general, more demanding than for direct preconditioners, but there are applications worth of doing this. Here we will deal with a specific algorithm that provides the approximate inverse factorization in the form $A \approx \tilde{Z}\tilde{Z}^T$. This type of factorization has been found useful not only in general parallel iterative solvers but also in solving generalized eigenvalue problems that arise from computation of electronic structures. See, for example, [6], [7] and the recent survey [14].

The computational scheme corresponds to the modified variant of the Gram–Schmidt process with a non-standard inner product induced by the

matrix A. Dropping in the algorithm that causes incompleteness of the factors has been originally proposed as the so-called *absolute dropping* that is based on a comparison of magnitudes of intermediate quantities with chosen dropping level, see [5], [2]. The idea of relative dropping is based on comparison of magnitudes of intermediate quantities with magnitudes of other entries that appear in the factorization. In the chosen approximate inverse factorization this has led to the adaptive dropping [12] that is theoretically sound as explained there. In special cases as for the M-matrices the process reflects bounds derived for to finite precision computations.

The rest of the paper is organized as follows. First we recall the generalized Gram–Schmidt process that is the algorithm behind the approximate inverse factorization. This process is combined with a class of permutation that allows to gain some theoretical insight into the factorization. New ways to the adaptive dropping are discussed in Section 3. The paper is concluded by numerical experiments and conclusions.

2 Generalized Gram–Schmidt process

It has been already shown that the generalized Gram–Schmidt process may provide robust preconditioners [5]. Here we will consider its modified version because it provides more favorable results with respect to the computational cost and stability [2]. Moreover, a column pivoting is used to combine the process with the above-mentioned permutation class. The resulting numerical scheme known as the left-looking approach is summarized in Algorithm 1. The inner product of vectors is denoted here by $\langle \cdot, \cdot \rangle_A$. Al-

Algorithm 1 Modified version of the Gram–Schmidt process with column permutation and with respect to the inner product $\langle \cdot, \cdot \rangle_A$

for $k := 1 \to n$ **do**
  $z_k^{(0)} := Pe_k$
  for $j := 1 \to k-1$ **do**
    $\alpha_{j,k} := \langle z_k^{(j-1)}, z_j \rangle_A$
    $z_k^{(j)} := z_k^{(j-1)} - \alpha_{j,k} z_j$
  end for
  $\alpha_{k,k} := \| z_k^{(k-1)} \|_A$
  $z_k := z_k^{(k-1)} / \alpha_{k,k}$
end for

gorithm 1 computes for each k a column z_k of the factor Z using the vector Pe_k that is A-orthogonalized against the previously computed vectors.

This process produces the upper triangular matrix $U = [\alpha_{i,j}]$ and the matrix $Z = [z_1, z_2, \ldots, z_n]$, such that $ZU = P$, where P is a permutation matrix. If $P = I$, Z is also upper triangular. Matrix U represents the Cholesky factor in the factorization of $P^T AP = U^T U$, and matrix Z represents the inverse factor in the factorization of $A^{-1} = ZZ^T$. As it is shown in [12], an absolutely necessary step that implies the practical usefulness is the pivoting based on magnitudes of the diagonal entries of U. The factorized matrix should be symmetrically permuted by a matrix permutation P chosen so that the entries of the factor U satisfy

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

$$\alpha_{i,i}^2 \geq \sum_{j=i}^{k} \alpha_{j,k}^2, \quad k = i+1, \ldots, n. \tag{3}$$

Note that (2) and (3) also imply

$$\alpha_{j,j} > |\alpha_{j,k}|, \quad j = 1, \ldots, n, \quad k = j+1, \ldots, n. \tag{4}$$

The permutation P in the above mentioned form is not known a priori and has to be computed on-the-fly. One possibility to get it is to use an additional computation of the A-orthogonalization coefficients using the classical variant of the Gram–Schmidt process [10] as proposed in [12]. This is done as follows. For each k and $j = k, \ldots, n$, the A-norms of the vectors $z_j^{(k-1)}$ are updated using the scheme

$$\|z_j^{(k-1)}\|_A^2 = \|z_j^{(k-2)}\|_A^2 - \langle z_j^{(0)}, z_{k-1}\rangle_A^2. \tag{5}$$

The new k-th column vector $Pe_k \equiv e_i$ is chosen such that

$$\|z_i^{(k-1)}\|_A = \max_{k \leq j \leq n} \|z_j^{(k-1)}\|_A. \tag{6}$$

Permutation P is thus obtained implicitly by the application of column pivoting with the criterion (6). As it is well-known, the generalized Gram–Schmidt process then computes the Cholesky factorization of $P^T AP$ and the triangular factorization of A^{-1} only in exact arithmetic. The error of the Cholesky factorization in finite precision arithmetic depends on the choice of the actual numerical scheme of the generalized Gram–Schmidt process. This is the reason that we need to distinguish between the modified and classical Gram–Schmidt schemes. Numerical properties of various implementations of the process in finite precision arithmetic are analyzed in [13] and [11].

As for the cost of the factorization, we need to distinguish its two components. The timings to construct Z are given in Table 2 in [5]. We can see that

on average the right-looking computation is slower with respect to the drop tolerance-based incomplete Cholesky factorization with a factor not more than two. Note that in the cost we need to consider only computation of the factor Z, the factor U is used only for theoretical purposes. In the dense case there is the asymptotic equivalence of the LU factorization and this algorithm for solving even general nonsymmetric systems; see [1], page 22. This means that typically more complex computation to get Z as mentioned above is caused mainly by more involved data structures and often larger amount of intermediate fill-in in the process. The pivoting is implemented via heaps and it is therefore very cheap. Incremental condition estimation is cheap as well. As for applying the factors within the conjugate gradients, this cost is formally proportional to the number of nonzeros in the factor. In addition, the fact that the applications is performed by matrix-vector multiplications means that the performance is highly scalable and avoid parallelization bottlenecks of direct incomplete factorizations. The overall efficiency may be strongly enhanced by exploiting block structure of the matrix. Let us mention, although this gets out of the scope of this paper, that this approximate inverse preconditioning leads in some applications to a nearly linear scaling of the block-based preconditioned iterative solver with respect to the problem size [6]; see also more details on an extension to the block approach in [3] and [4].

3 Approximate inverse preconditioning employing adaptive dropping

In order to obtain a sufficiently sparse approximate inverse preconditioning, some quantities have to be dropped, i.e., truncated to zero. In our approximate computations we use an extra upper tilde notation, e.g., $\tilde{Z}$, $\tilde{z}_k$. Without going into details we start with the result developed in [12]. Based on a theoretical understanding of the Cholesky factorization of M-matrices, an adaptive dropping parameter (level) has been introduced at k-th step for the computation of z_k, in the form

$$\boldsymbol{\tau_k} \leq \frac{\boldsymbol{\tau}}{\kappa(\tilde{U}_k)}, \tag{7}$$

where $\boldsymbol{\tau}$ denotes the *accuracy baseline* chosen for the decomposition and where $\tilde{U}_k \in \mathbb{R}^{k\times k}$ denotes the principal leading submatrix of the matrix $\tilde{U}$. Symbol $\kappa(\tilde{U})$ denotes the condition number defined by the singular values of $\tilde{U}$.

In finite precision arithmetic, the quality of the factors U and Z strongly depends on the departure of orthogonality caused by rounding errors. This principle can be up to some extent transferred into the incomplete process and this is the basis for the approach discussed in [12]. The formula (7) points out

that the growth of $\kappa(\tilde{U}_k)$ with respect to k can significantly influence size of magnitudes of the entries dropped within k-th step.

It has been shown in [12] that the generalized Gram–Schmidt process with pivoting based on rule (6) combined with a specific adaptive dropping (7) may lead to successful preconditioner. Here the success of the preconditioner is studied using criterion that couples the sparsity and convergence of the preconditioned conjugate gradient method. There are several ways to determine $\kappa(\tilde{U}_k)$ exactly or approximately. Some of them are

a) singular value decomposition of $\tilde{U}_k$ (SVD),

b) fraction of the extremal diagonal entries given by $\kappa(\tilde{U}_k) \approx \frac{\max(\mathrm{diag}(\tilde{U}_k))}{\min(\mathrm{diag}(\tilde{U}_k))} = \kappa(\tilde{D}_k)$ or

c) incremental condition estimator (condest($\tilde{U}_k$)).

The singular value decomposition represents the most accurate approach, but it also the most time demanding method and cannot be typically used in large sparse computations. Further, the estimates based on the fraction of the two extremal diagonal entries that we denote using symbol $\kappa(\tilde{D}_k)$, where $\tilde{D}_k$ represents the diagonal part of $\tilde{U}_k$, i.e., $\tilde{D}_k = \mathrm{diag}(\tilde{U}_k)$ are rather simple and easy to implement using a heap for the diagonal entries. But this approach is not a good choice, in general. However, the inequality (2) implies that the eigenvalues of the factor $\tilde{U}_k$ are sorted by their magnitudes. Moreover, the inequality (3) determines, that column vectors of the trailing principal submatrices of U having the largest Euclidean norm are concentrated in the first column. Therefore the estimate of $\kappa(\tilde{U}_k)$ based on $\kappa(\tilde{D}_k)$ may be for our pivoted algorithm a better choice than for a non-pivoted algorithm. The estimate based on an incremental condition estimator seems to offer a compromise between the previously mentioned approaches. We believe that it can represent a method of choice in some cases and its influence is studied here experimentally. A specific feature is that the approximate inverse factorization enables to consider the incremental condition estimator in the Euclidean norm from [9] that exploits both the direct and inverse factors and it is shown to be better than standard incremental condition estimators. In particular, the authors in [9] show that one can construct a highly accurate incremental condition number estimator in this norm. In our case of the approximate inverse factorization, both factors are known because $\tilde{U}^{-1} \approx P^T \tilde{Z}$ and we can thus directly use this estimator. The differences among the mentioned approaches to compute or estimate $\kappa(\tilde{U}_k)$ will be discussed in the subsequent section.

4 Numerical experiments

We have chosen several problems from the University of Florida collection [8], see Table 1. In particular, we have chosen smaller problems in order to compare all the above mentioned methods for computing $\kappa(\tilde{U}_k)$. Iteration counts of the preconditioned CG is denoted as it, an extra upper subscript determines the type of the approximation of $\kappa(\tilde{U}_k)$. Table 1 gives in the column it the minimum and maximum iteration counts for the three considered condition number estimates varying values of $\boldsymbol{\tau}$. The values are separated by a dash as, e.g., $\min(it) - \max(it)$. We believe that this can better show advantages and disadvantages of various approaches. In a detail, we study

Table 1: The test matrices.

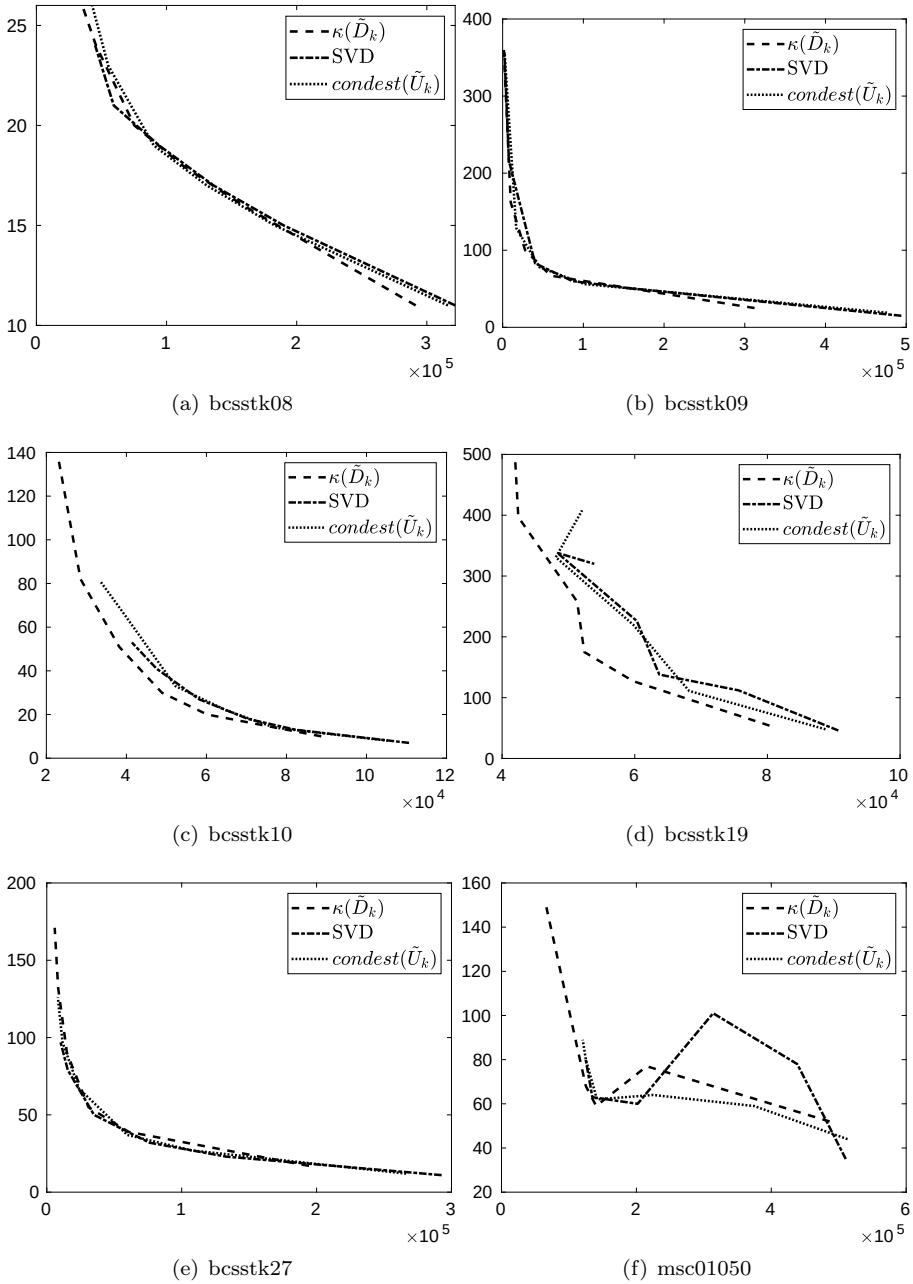
Matrix	Dimension $[n]$	nnz(A)	$it^{\kappa(\tilde{D}_k)}$	it^{SVD}	$it^{\mathrm{condest}(\tilde{U}_k)}$
bcsstk08	1 076	12 960	11 – 26	11 – 24	11 – 26
bcsstk09	1 083	18 437	25 – 361	15 – 358	19 – 354
bcsstk10	1 086	20 070	10 – 139	7 – 53	8 – 81
bcsstk19	817	6 853	54 – 500	46 – 338	48 – 408
bcsstk27	1 224	56 126	17 – 171	11 – 97	12 – 126
msc01050	1 050	26 198	52 – 149	35 – 101	44 – 89
nos2	957	4 137	9 – 500	1 – 121	5 – 146
nos3	960	15 884	20 – 246	13 – 119	15 – 171
nos7	729	4 617	8 – 34	6 – 28	7 – 31

the preconditioner behavior in the terms of nnz$(\tilde{Z})$, it, and their connection to $\boldsymbol{\tau}$. In addition, we are interested in sensitivity of the choice of $\boldsymbol{\tau}$, i.e., how difficult it is to determine a suitable value of $\boldsymbol{\tau}$ that leads to a reasonably fast convergence with sufficiently sparse factor $\tilde{Z}$. We consider the drop tolerance baselines $\boldsymbol{\tau} = 0.01, 0.05, 0.10, 0.20, 0.40, 0.60$. Figures 1 and 2 depict the iteration counts of the preconditioned CG as a function of nnz$(\tilde{Z})$ for all matrices from Table 1. It is easy to see that one of the (favorable) properties of the all considered approaches is nearly monotonic relation between nnz$(\tilde{Z})$ and number of iteration of preconditioned CG. The approximation of $\kappa(\tilde{U}_k)$ based on $\kappa(\tilde{D}_k)$ represents the best choice for several problems. We can see that, for example, this approach does not work well for the matrix nos2. Here the difference between the quantities $\max(it^{\kappa(\tilde{D}_k)})$ and $\min(it^{\kappa(\tilde{D}_k)})$ is rather large. In general, finding a suitable value of $\boldsymbol{\tau}$ may be sometimes difficult and a trial and error strategy has to be employed. Good dropping should avoid this as much as possible. Adaptive dropping based on SVD leads for given values of

accuracy $\boldsymbol{\tau}$ to the very dense factors $\tilde{Z}$. On the other hand, the corresponding $\min(it^{\mathrm{SVD}})$ is minimal with respect to other approaches and also difference between $\max(it^{\mathrm{SVD}})$ and $\min(it^{\mathrm{SVD}})$ is also significantly better than in previous case. Incremental condition estimator exhibits similar behavior as using SVD. As one can see these two strategies provide similar results. But since the incremental condition is cheap it can be considered a method of choice.

5 Summary

Based on the theoretical results developed for the generalized Gram–Schmidt process in finite precision arithmetic it has been shown that an adaptive dropping rule can be based on the condition number of the leading principal submatrices in the Cholesky factor [12]. We have studied three different approaches to estimate the condition numbers of the leading principal submatrices of the direct factor and used them in the formula for adaptive dropping. The estimates based on SVD and condest($\tilde{U}_k$) are very close and both lead to robust preconditioners. One of the most favorable properties of these two approaches is the observed robustness with respect to the chosen dropping baseline of the preconditioner. The adaptive dropping rule based on SVD or condest($\tilde{U}_k$) leads to nearly parameter-free computation of the preconditioner, i.e., preconditioners work well for a wide range of $\boldsymbol{\tau}$. Moreover, condest($\tilde{U}_k$) does not have any limitation arising from the dimension of the problem. The estimate based on $\kappa(\tilde{D}_k)$ leads for several cases to the most efficient preconditioning, but trial and error strategy to find a suitable value of $\boldsymbol{\tau}$ must be applied for some problems. On the other hand, it could be a method of choice for time dependent problems where matrix A does not change much and some stable value of $\boldsymbol{\tau}$ is known. In general, the estimate condest($\tilde{U}_k$) represents a good compromise delivering predictable behavior of the preconditioner (in terms of convergence) without the use of trial and error strategy obtaining suitable $\boldsymbol{\tau}$ in a majority of the test cases.

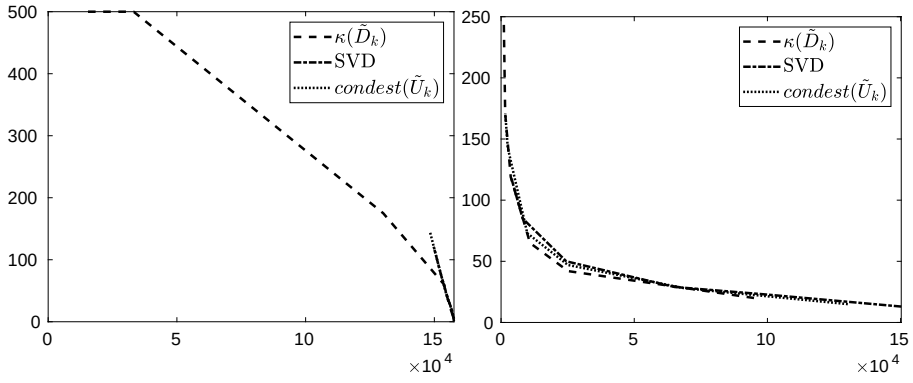


(a) bcsstk08 (b) bcsstk09

(c) bcsstk10 (d) bcsstk19

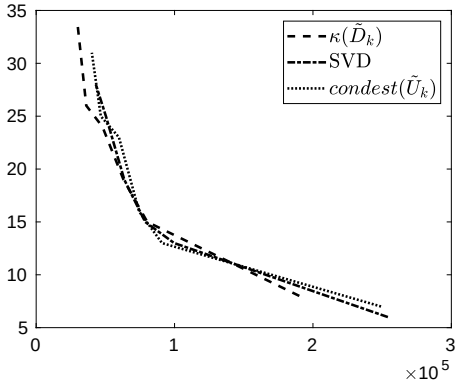
(e) bcsstk27 (f) msc01050

Figure 1: Convergence of PCG in terms of number of iterations as a function of sparsity of the preconditioner $\mathrm{nnz}(\tilde{Z})$.



(a) nos2

(b) nos3



(c) nos7

Figure 2: Convergence of PCG in terms of number of iterations as a function of sparsity of the preconditioner $\mathrm{nnz}(\tilde{Z})$.

References

[1] M. Benzi. A Direct Row-Projection Method for Sparse Linear Systems. PhD. Thesis. *Department of Mathematics. North Carolina State University, Raleigh, NC,* 1993.

[2] 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.

[3] M. Benzi, R. Kouhia, and M. Tůma. An assessment of some preconditioning techniques in shell problems. *Communications in Numerical Methods in Engineering*, 14:897–906, 1998.

[4] M. Benzi, R. Kouhia, and M. Tůma. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Comput. Methods Appl. Mech. Engrg.*, 190(49-50):6533–6554, 2001.

[5] 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.

[6] M. Challacombe. A simplified density matrix minimization for linear scaling self-consistent field theory. *J. Chem. Phys.*, 110:2332–2342, 1999.

[7] M. Challacombe. A general parallel sparse-blocked matrix multiply for linear scaling scf theory. *Comp. Phys. Comm.*, 128:93–107, 2000.

[8] T. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38 (1):1–25, 2011.

[9] J. Duintjer Tebbens and M. Tůma. On incremental condition estimators in the 2-norm. *SIAM J. Matrix Anal. Appl.*, 35(1):174–197, 2014.

[10] N. J. Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

[11] J. Kopal. *Generalized Gram–Schmidt Process: Its Analysis and Use in Preconditioning*. PhD thesis, Technical University in Liberec, 2014.

[12] J. Kopal, M. Rozložník, and M. Tůma. Factorized approximate inverses with adaptive dropping. *SIAM J. Sci. Comput.*, 38(3):A1807–A1820, 2016.

[13] 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.

[14] E. H. Rubensson, A. G. Artemov, A. Kruchinina, and E. Rudberg. Localized inverse factorization. arXiv:1812.04919.

Jiří Kopal,
Technical University of Liberec, Institute of Novel Technologies and Applied Informatics, Studentská 1402/2, 461 17 Liberec 1, Czech Republic,
Email: jiri.kopal@tul.cz

Miroslav Rozložník,
Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic,
Email: miro@cs.cas.cz

Miroslav Tůma,
Faculty of Mathematics and Physics, Charles University in Prague, Sokolovská 83, 186 75 Praha 8,
Email: mirektuma@karlin.mff.cuni.cz