Approximate inverse preconditioners with adaptive dropping *

Jiří Kopal†, Miroslav Rozložník, Miroslav Tůma‡

July 4, 2013

Keywords: approximate inverse, Gram–Schmidt orthogonalization, incomplete decomposition, preconditioned iterative method.

Abstract

It is well-known that analysis of incomplete Cholesky and LU decompositions with a general dropping is very difficult and of limited applicability, see, for example, the results on modified decompositions [1], [2], [3] and later results based on similar concepts. This is true not only for the dropping based on magnitude of entries but it also applies to algorithms that use a prescribed sparsity pattern.

This paper deals with dropping strategies for a class of AINV-type incomplete decompositions [4] that are based on the generalized Gram–Schmidt process. Its behavior in finite precision arithmetic has been discussed in [5]. This analysis enables better understanding of the incomplete process, and the main goal of the paper is to propose a new adaptive dropping strategy and to illustrate its efficiency for problems in structural mechanics.

1 Introduction

Iterative methods are widely used for the solution of large sparse linear systems

$$Ax = b, \tag{1}$$

where A is the $n \times n$ system matrix, x is the vector of unknowns and b is the right-hand-side. A large source of such systems are discretizations of partial differential

*Partially supported by the projects P201/13-06684S and 108/11/0853 of the Grant Agency of the Czech Republic.

†Technical University of Liberec, Department of Mathematics, Liberec (jiri.kopal@tul.cz)

‡Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic, (miro@cs.cas.cz, tuma@cs.cas.cz).

equations that arise in many applications in science and engineering and that often provide sparse matrices. In order to increase the robustness of iterative methods, system (1) should be transformed by preconditioning. In this paper we will consider only symmetric and positive definite systems. There are two basic general purpose classes of preconditioners [6] for the conjugate gradient method, that is the method of choice here. Namely, incomplete Cholesky factorizations and factorized approximate inverses. Both of them can be value-based in which the locations of permissible fill entries are determined together with the numerical factorization of A; the entries in the computed factors or intermediate quantities that exceed a prescribed threshold are *dropped*. The success of such approach often depends on being able to choose a suitable threshold, and this can be highly problem dependent.

Approximate inverse preconditioners are based on representing the inverse A^{-1} via incomplete decomposition $\tilde{Z}\tilde{Z}^T$ with the upper triangular factor $\tilde{Z}$. The system (1) preconditioned from both sides can be then written as

$$\tilde{Z}^T A \tilde{Z} y = \tilde{Z}^T b, \qquad x = \tilde{Z} y. \tag{2}$$

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. 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 [5] motivates us to find tools useful for decision which nonzero fill-in entries do not improve this accuracy. Consequently, such entries can be dropped. We will illustrate that their detection can be based on monitoring such intermediate quantities as the loss of A-orthogonality between the column vectors in the factor $\tilde{Z}$ and the conditioning of the factor $\tilde{U}$ (that represents an approximation to the transpose of the Cholesky factor of A). We will attempt to keep the conditioning $\tilde{U}$ as low as possible by combining the dropping strategy with pivoting that approximately minimizes the estimate for $\kappa(\tilde{U})$. Based on the results for some test matrices from structural engineering, we will see that this approach leads to rather powerful and still sparse preconditioners. Note that the loss of A-orthogonality $\|\tilde{Z}^T A \tilde{Z} - I\|$ as well as the norm $\|A^{-1} - \tilde{Z}\tilde{Z}^T\|$ play a similar role as the quantities $\|I - \tilde{L}^{-1} A \tilde{L}^{-T}\|$ and $\|A - \tilde{L}\tilde{L}^T\|$ in the incomplete Cholesky factorization, see [7].

This paper represents a continuation of the research published in [8]. The analysis has been refined and the accompanying software was extended. The paper is organized as follows. In Section 2 we present basics of the underlying theory. Section 3 presents the scheme of our approach called *a posteriori filtering* with experimental results. We consider our contribution being a step on the way to construct incomplete decompositions with a better understanding of the dropping process.

2 The generalized Gram–Schmidt process

The term *generalized Gram–Schmidt process* is used here to denote the orthogonalization with the non–standard inner product induced by a symmetric and positive definite matrix A. The column vectors of the matrix $Z^{(0)} = [z_1^{(0)}\ z_2^{(0)}\ \ldots\ z_n^{(0)}]$ (initial basis) are orthogonalized against previously computed columns in Z. In exact arithmetic, all variants of the generalized Gram–Schmidt process lead to the factors Z and U satisfying $Z^{(0)} = ZU$ such that $Z^T AZ = I$ and

$$(Z^{(0)})^T AZ^{(0)} = U^T U \text{ (Cholesky factorization)}, \tag{3}$$

$$A^{-1} = ZZ^T \text{ (inverse triangular factorization)}. \tag{4}$$

The generalized Gram–Schmidt process is summarized in Algorithm 1 where $Z = [z_1\ z_2\ \ldots\ z_n]$ and $U = [u_{j,i}]$. Note that here we consider the *modified* version of the generalized Gram–Schmidt process.

Algorithm 1 Modified version of the generalized Gram–Schmidt process

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

In finite precision arithmetic, the identities (3) and (4) are no longer valid. In the following, we will use the notation with bar (e.g. $\bar{Z}, \bar{U}$) for the quantities computed in finite precision arithmetic with no other dropping (i.e. for the complete Gram–Schmidt process). The bounds for the quantities $\|Z^{(0)} - \bar{Z}\bar{U}\|$, $\|\bar{Z}^T A\bar{Z} - I\|$ and $\|(Z^{(0)})^T AZ^{(0)} - \bar{U}^T\bar{U}\|$ for the main versions of the generalized Gram–Schmidt process have been derived in [5]. Even in the presence of rounding errors these algorithms produce the matrices $\bar{Z}$ and $\bar{U}$ that are good approximations to the identities (3) and (4). But such computation is inevitably time consuming because it leads to rather dense factors. For preconditioning we have to use their incomplete versions. In order to distinguish we will use the notation with tilde (e.g. $\tilde{Z}, \tilde{U}$) for computed quantities in incomplete decompositions with appropriate dropping.

As noted above, 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 individual algorithm possibly taking into account also some target computational architecture. It has been shown in [4] 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 optimum drop tolerance

may be difficult. Up to now, there has not been proposed a better way to find it than by trial-and-error.

In the following we propose a new approach which turns out to be a safe and efficient strategy. We assume that the algorithm is started with the initial vector basis $Z^{(0)} = I$. In the numerical experiments presented here we use only the matrix *bcsstk07* from structural mechanics (see the MatrixMarket collection). It has a small dimension 420 that enables us to show very detailed results. However, we have obtained very similar results for the whole group of matrices arising from other engineering problems.

3 Adaptive dropping by a posteriori filtering

Here we will deal with the *incomplete* version of the generalized Gram–Schmidt process that was introduced above. The algorithm is based on dropping nonzero entries less than an *adaptive* drop tolerance τ_i prescribed in individual major steps i of the process. The strategy explained below is motivated by a natural demand to get the residual norms of the approximately computed columns $\tilde{z}_i$ in $\tilde{Z}_i$ on the same level of accuracy. As a theoretical motivation, we consider the *complete* Gram–Schmidt process, that is the process computed in the floating-point arithmetic without any dropping. Having the initial basis $Z^{(0)} = I$, its i*-th* step provides the leading principal submatrices $\bar{U}_i$ and $\bar{Z}_i$, such that

$$\bar{U}_i \bar{Z}_i - I_i = \Delta E_i, \quad \Delta E_i = [\delta e_1 \; \delta e_2 \; \ldots \; \delta e_i] \tag{5}$$

for some residual matrix ΔE_i. Based on our experiments and without going into details we assume that

$$\|\Delta E_i\| \leq \epsilon \, \mathcal{O}(i^{3/2}) \|\bar{U}_i\| \|\bar{Z}_i\| \tag{6}$$

Consider the i*-th* column residuals of the vectors $\tilde{z}_i$ and $\hat{z}_i$ in the form

$$\tilde{U}_i \tilde{z}_i - e_i \qquad \text{and} \qquad \tilde{U}_i \hat{z}_i - e_i, \tag{7}$$

where the i*-th* vector $\tilde{z}_i$ is the resulting vector obtained from $\hat{z}_i$ after dropping and $\hat{z}_i$ was evaluated using previously computed vectors of $\tilde{Z}$ as

$$\hat{z}_i = \frac{z_i^{(0)} - \sum_{k=1}^{i-1} \tilde{\alpha}_{ki} \tilde{z}_k}{\| z_i^{(0)} - \sum_{k=1}^{i-1} \tilde{\alpha}_{ki} \tilde{z}_k \|_A} = \frac{\hat{z}_i^{(i-1)}}{\| \hat{z}_i^{(i-1)} \|_A}. \tag{8}$$

This formula does not consider the roundoff error that should be typically much smaller than any explicit dropping. Let us denote by $\delta z_i = \tilde{z}_i - \hat{z}_i$ the error from this additional dropping. In this way we get

$$\delta z_i = \tilde{U}_i^{-1} \left[(\tilde{U}_i \tilde{z}_i - e_i) - (\tilde{U}_i \hat{z}_i - e_i) \right]. \tag{9}$$

Naturally, the perturbation δz_i caused by the dropping should not exceed the residual norm multiplied by the norm of $\tilde{U}_i^{-1}$ in its magnitude. Further, we assume that both $\|\tilde{U}_i\tilde{z}_i - e_i\| \leq \tau_i\|\tilde{U}_i\|\|\hat{z}_i\|_\infty$ and $\|\tilde{U}_i\hat{z}_i - e_i\| \leq \tau_i\|\tilde{U}_i\|\|\hat{z}_i\|_\infty$. This corresponds to the assumption for the floating-point process in (6), but it also corresponds to the perturbation caused just by evaluating the residual (7).

Putting everything together and asking for a uniform bound enforced by a given parameter $\tau < 1$ we have

$$\|\delta z_i\|_\infty \leq \tau_i\|\tilde{U}_i\|\|\tilde{U}_i^{-1}\|\|\hat{z}_i\|_\infty. \tag{10}$$

In this way we get

$$\frac{\|\delta z_i\|_\infty}{\|\hat{z}_i\|_\infty} \approx \tau_i \leq \frac{\tau}{\kappa(\tilde{U}_i)}. \tag{11}$$

As we can see, keeping the uniform bound on the residual norms leads to the adaptive dropping in the decomposition steps with the drop tolerance τ_i. This new dropping technique based on monitoring the condition number for $\tilde{U}_i$ will be called the *a posteriori filtering*. Further, the singular values interlacing property [9] $\kappa(\tilde{U}_1) \leq \kappa(\tilde{U}_2) \leq \cdots \leq \kappa(\tilde{U}_i) \leq \cdots \leq \kappa(\tilde{U}_n)$ implies that the sequence of drop tolerances τ_i is non-increasing. Typically, the relative error $||\delta z_i||_\infty/||\hat{z}_i||_\infty$ should decrease as $\kappa(\tilde{U}_2)$ increases. Note that the proposed dropping strategy does not depend on the conditioning of the whole problem, but only on the local conditioning. Therefore, a natural practical strategy is to keep the increase in the sequence of the local condition numbers $\kappa(\tilde{U}_i)$ as small as possible. As we will see later, this can be achieved by appropriate preprocessing.

The a posteriori filtering algorithm is put down in Matlab-like notation as Algorithm 2. It is clear from (11) that normalizing of the system matrix before dropping is not necessary because the dropping is scaled internally. The mask is the vector of boolean variables. All nonzeros in the vector $\hat{z}_i^{(i-1)}$ smaller than τ_i multiplied by $\|\hat{z}_i^{(i-1)}\|_\infty$ are marked as *zero* in the $mask$ vector (the first line). Second line in the algorithm safeguards the diagonal entries. In examples presented here we will see that the a posteriori filtering seems to be especially very suitable for sequences of matrices $\tilde{U}_i$ where $\kappa(\tilde{U}_i)$ grows slowly with i. Algorithm 1 extended by the a posteriori filtering (Algorithm 2) is put down here as Algorithm 3. There is a lot of ways to estimate $\kappa(\tilde{U}_i)$. Here we use the condition number estimate based on the fraction of the maximum over the minimum diagonal entry of $\tilde{U}_i$.

Algorithm 2 A posteriori filtering based on conditioning of $\tilde{U}_k$

$mask := |\hat{z}_i^{(i-1)}| \geq \frac{\tau\|\hat{z}_i^{(i-1)}\|_\infty}{\kappa(\tilde{U}_i)}$
$mask_i := 1$
$\tilde{z}_i^{(i-1)} := \hat{z}_i^{(i-1)} .* mask$

Figure 1 demonstrates that CG achieves a good convergence rate for all considered values of τ. Nevertheless, the preconditioner may be still considered rather dense

Algorithm 3 Modified generalized GS with a posteriori filtering

for $i := 1 \to n$ **do**
  for $j := 1 \to i-1$ **do**
    $\tilde{u}_{j,i} := \langle \hat{z}_i^{(j-1)}, \tilde{z}_j \rangle_A$
    $\hat{z}_i^{(j)} := \hat{z}_i^{(j-1)} - \tilde{u}_{j,i} \tilde{z}_j$
  end for
  $\tilde{u}_{i,i}^{(0)} := \langle \hat{z}_i^{(i-1)}, \hat{z}_i^{(i-1)} \rangle_A^{1/2}$
  $mask := |\hat{z}_i^{(i-1)}| \geq \tau \|\hat{z}_i^{(i-1)}\|_\infty / \kappa(\tilde{U}_i)$
  $mask_i := 1$
  $\tilde{z}_i^{(i-1)} := \hat{z}_i^{(i-1)} .* mask$
  $\tilde{u}_{i,i} := \langle \tilde{z}_i^{(i-1)}, \tilde{z}_i^{(i-1)} \rangle_A^{1/2}$
  $\tilde{z}_i := \tilde{z}_i^{(i-1)} / \tilde{u}_{i,i}$
end for

as we can see the nonzero count $nnz(\tilde{Z})$. The reason for this effect is revealed in Figure 2 depicting fast growth of the condition number of $\tilde{U}_i$ as well as the relatively dense sparsity pattern of $\tilde{Z}$. Motivated by this result, the subsequent part of the paper proposes a way to obtain sparser preconditioners, but let us remind that the structural and accuracy effects are typically coupled.

4 A posteriori filtering and matrix reordering

An important tool that may help to make preconditioners sparser and often even more useful is their preprocessing. We will show that a specific matrix reordering may significantly improve the computed matrix $\tilde{Z}$ in this respect. We will choose the reordering such that it approximately minimizes the growth of the conditioning of the leading principal submatrices $\tilde{U}_i$. We will demonstrate that this approach is able to provide factor $\tilde{Z}$ such that the partial decomposition captures largest eigenvalues of A first, see the concept of partial factorization in [10]. In order to estimate a potential of this approach, consider the permuted matrix $\mathring{A} = P^T A P = \mathring{U}^T \mathring{U}$, $\mathring{U} = [\mathring{u}_{i,j}]$, where P is a permutation matrix, such that the entries of $\mathring{U}$ hold following inequalities

1. $\mathring{u}_{i,i}^2 \geq \sum_{k=i}^{j} \mathring{u}_{k,j}^2, \quad \forall j = i+1, \ldots, n,$
2. $\mathring{u}_{1,1} \geq \mathring{u}_{2,2} \geq \ldots \geq \mathring{u}_{n,n}$,
3. $\mathring{u}_{i,i} > |\mathring{u}_{i,j}|, \quad \forall j = i+1, \ldots, n.$

Figures 3 and 4 for our permuted matrix show that the convergence rate of CG have similar properties as above but $nnz(\tilde{Z})$ has been significantly reduced. Clearly, this reordering improves the curve of the conditioning of $\tilde{U}_i$, and together with the a posteriori filtering it enables to drop safely much more entries. The situation is

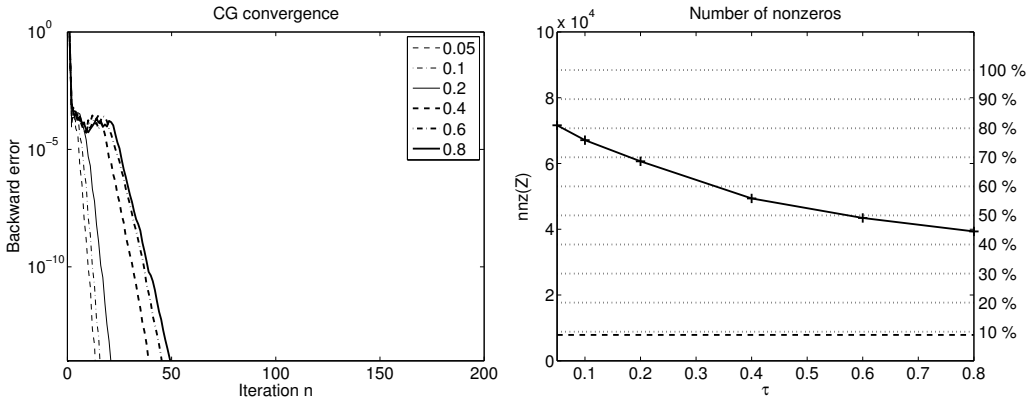

Figure 1: Figures corresponding to the Algorithm 3. Figure on the *left* depicts convergence curves of CG preconditioned by $\tilde{Z}$ for a few values of τ. Figure on the *right* shows dependency on the nonzero count in the $\tilde{Z}$ factor on τ. *Dotted horizontal lines* denote ratio of the nonzero counts with respect to a dense triangular factor ($(n^2+n)/2$ nonzero entries). *Dashed horizontal line* highlights the number of nonzeros in A ($nnz(A)$).

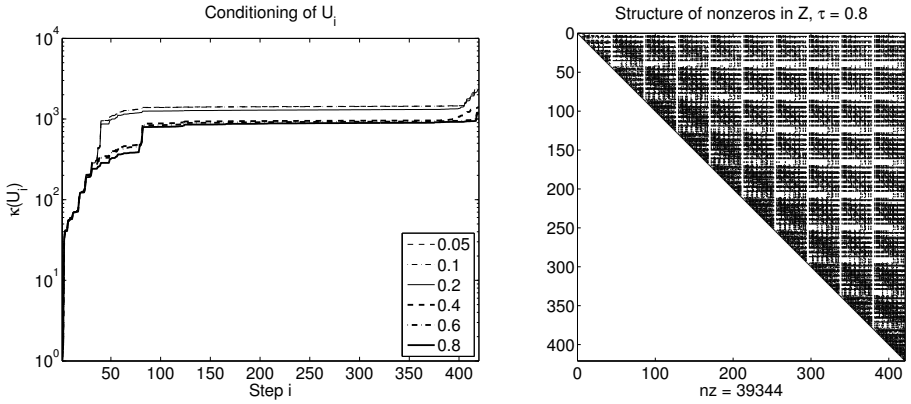

Figure 2: Figures corresponding to the Algorithm 3. Figure on the *left* depicts dependency of the conditioning of the leading principal submatrices $\tilde{U}_i$ on i. Figure on the *right* shows the sparsity structure of the preconditioner for the largest considered $\tau = 0.8$.

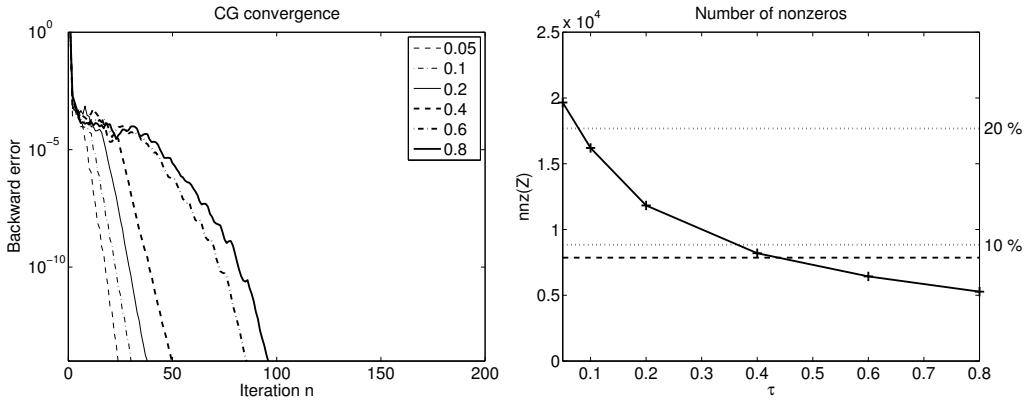

Figure 3: Figures corresponding to the Algorithm 3 on the permuted matrix $\hat{A}$. Figure on the *left* depicts convergence curves of CG preconditioned by $\tilde{Z}$ for a few values of τ. Figure on the *right* shows the dependency of nonzero counts in the $\tilde{Z}$ factor on τ. *Dotted horizontal lines* denotes ratio of the nonzero count related to the dense triangular factor ($(n^2+n)/2$ nonzero entries). *Dashed horizontal line* highlights the nonzero count in the matrix A ($nnz(A)$).

apparent from Figures 2 and 4. The structure of nonzeros in $\tilde{Z}$ has a specific shape as it often happens when using this type of reordering. In particular, the first few of principal leading submatrices $\tilde{Z}_i$ have very sparse (nearly diagonal) pattern. The exact reordering of this kind is expensive but in the next section we will present its computationally efficient variant based on approximate pivoting.

5 A posteriori filtering and pivoting

In order to get a practical strategy, the expensive static reordering from the previous subsection cannot be used. We will demonstrate here that it is possible to replace the a priori reordering by a pivoting performed in the course of the incomplete decomposition. We propose an approach based on a specific combination of the left-looking algorithm and the Cholesky factorization, although a right-looking implementation of the approximate inverse decomposition may be still used [4]. Formula (12) shows how the computed vector z_i can be used to update the A–norms of the vectors, which were not yet A–orthogonalized (here it is described, for simplicity, in exact arithmetic).

$$\|z_i^{(k)}\|_A^2 = (u_{i,i}^{(k)})^2 = a_{i,i}^2 - \sum_{j=1}^{k} \langle z_i, z_j^{(0)} \rangle_A^2, \quad k = 0, \ldots, i-1, \quad (u_{i,i}^{(i-1)})^{1/2} = u_{i,i} \tag{12}$$

In this way, the pivoting is easy and cheap to compute. For the *incomplete* algorithm this precomputation of pivots (12) may lead to a useful reordering as we demonstrate later.

The incomplete Algorithm 3 extended by the Cholesky-based *approximate* piv-

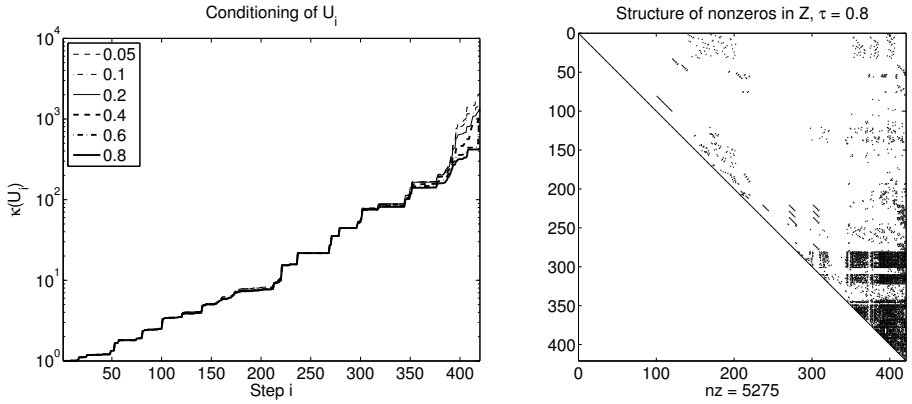

Figure 4: Figures corresponding to the Algorithm 3 applied to the permuted matrix $\hat{A}$. Figure on the *left* depicts dependency of the conditioning of the leading principal submatrices $\tilde{U}_i$ on i. Figure on the *right* shows the sparsity structure of the preconditioner for the largest considered $\tau = 0.8$.

Algorithm 4 Modified generalized GS with a posteriori filtering and Cholesky based pivoting

$Z^{(0)} = I$
$d^{(1)} := \text{diag}(A)$
for $i := 1 \to n-1$ **do**
  if $i > 1$ **then**
    for $j := i \to n$ **do**
      $d_j^{(i)} = d_j^{(i-1)} - \langle \tilde{z}_{i-1}, z_j^{(0)} \rangle_A^2$
    end for
  end if
  $k = \underset{i \le l \le n}{\text{argmax}}(d_l^{(i)})$
  $\text{swap\_columns}_{i,k}(Z^{(0)})$
  $\text{swap\_entries}_{i,k}(d^{(i)})$
  for $j := 1 \to i-1$ **do**
    $\tilde{u}_{j,i} := \langle \hat{z}_i^{(j-1)}, \tilde{z}_j \rangle_A$
    $\hat{z}_i^{(j)} := \hat{z}_i^{(j-1)} - \tilde{u}_{j,i} \tilde{z}_j$
  end for
  $\tilde{u}_{i,i}^{(0)} = \langle \hat{z}_i^{(i-1)}, \hat{z}_i^{(i-1)} \rangle_A^{1/2}$
  $mask = |\hat{z}_i^{(i-1)}| \ge \tau \|\hat{z}_i^{(i-1)}\|_\infty / \kappa(\tilde{U}_i)$
  $k = \underset{1 \le l \le n}{\text{argmax}}([Z^{(0)}]_{l,i})$
  $mask_k = 1$
  $\tilde{z}_i^{(i-1)} = \hat{z}_i^{(i-1)} .* mask$
  $\tilde{u}_{i,i} = \langle \tilde{z}_i^{(i-1)}, \tilde{z}_i^{(i-1)} \rangle_A^{1/2}$
  $\tilde{z}_i = \tilde{z}_i^{(i-1)} / \tilde{u}_{i,i}$
end for

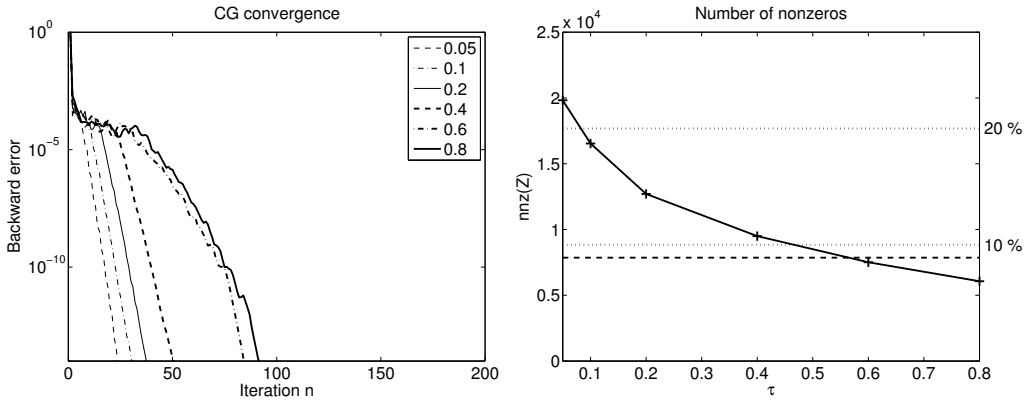

Figure 5: Figures corresponding to the Algorithm 4. Figure on the *left* depicts convergence curves of CG preconditioned by $\tilde{Z}$ for a few values of τ. Figure on the *right* shows dependency the of nonzeros in the $\tilde{Z}$ factor on τ. *Dotted horizontal lines* denotes percentage of the nonzeros with respect dense triangular factor ($(n^2+n)/2$ nonzero entries). *Dashed horizontal line* highlights number of nonzeros in the matrix A ($nnz(A)$).

oting is put down as Algorithm 4. The vector $d^{(j)}$ contains the *approximately* updated diagonal entries used as pivots. The formula $k = \underset{i \le l \le n}{\operatorname{argmax}}(d_l^{(i)})$ finds the pivot entry having the largest magnitude in the i*-th* step. The *swap* operations interchange corresponding values with respect to the position of the pivot. Moreover, $\text{swap\_columns}_{i,k}(Z^{(0)})$ swaps also the indices of the vectors $z_i^{(0)}$ and $z_k^{(0)}$. The index k obtained as $k = \underset{1 \le l \le n}{\operatorname{argmax}}([Z^{(0)}]_{l,i})$ denotes the position of the pivot in the vector $z_i^{(0)}$. The permutation introduced in the algorithm can be interpreted as running the generalized Gram–Schmidt process with $Z^{(0)}$ equal to the corresponding permutation matrix. The permutation is not known a priori, and $Z^{(0)}$ is then computed *on-the-fly* using (12). Consequently, $\tilde{Z}$ does not need to have the upper triangular form. Note that in our figures showing nonzero pattern we then deal with $(Z^{(0)})^T\tilde{Z}$ which is for the non-pivoted Algorithm 3 trivially equal to $\tilde{Z}$.

The Algorithm 4 can be implemented efficiently in the fully sparse mode [4]. Figures 5 and 6 show that the results are qualitatively the same as when using a priori static reordering. In particular, the results seem to be good from many possible points of view: iteration count, $nnz(\tilde{Z})$, limiting the work in the most crucial part to just a limited number of a dense vectors. Note that our results for the chosen matrix represent qualitatively the same results for the whole class of matrices from structural mechanics. The fact that a posteriori filtration with dynamic pivoting is able to improve approximation properties of $\tilde{Z}^T A\tilde{Z}$ significantly is demonstrated in Figure 7. It is clear that the strategy of this Section spreads the spectrum in the subsequent steps of the decomposition much less. The results point out that the theoretically motivated a posteriori filtration with pivoting leads to a rather powerful approximate inverse preconditioning.

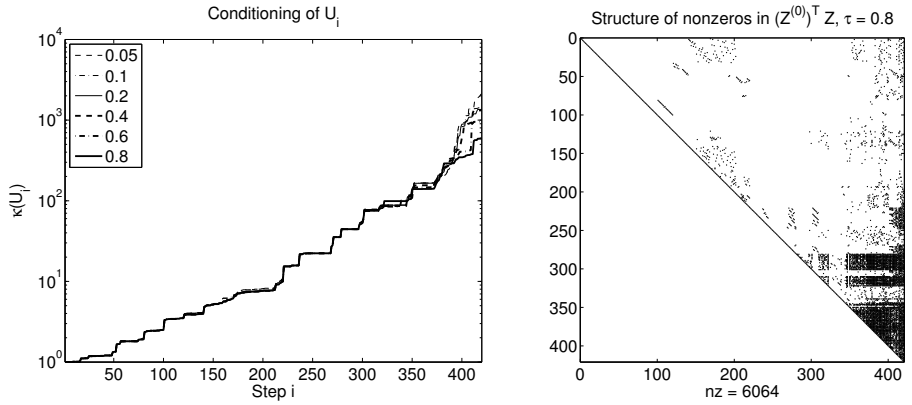

Figure 6: Figures corresponding to the Algorithm 4 on reordered matrix A. Figure on the *left* depicts dependency of the conditioning of the leading principal submatrices $\tilde{U}_i$ on i. Figure on the *right* shows the number of nonzeros in the matrix of the preconditioner for the largest cosidered $\tau = 0.8$.

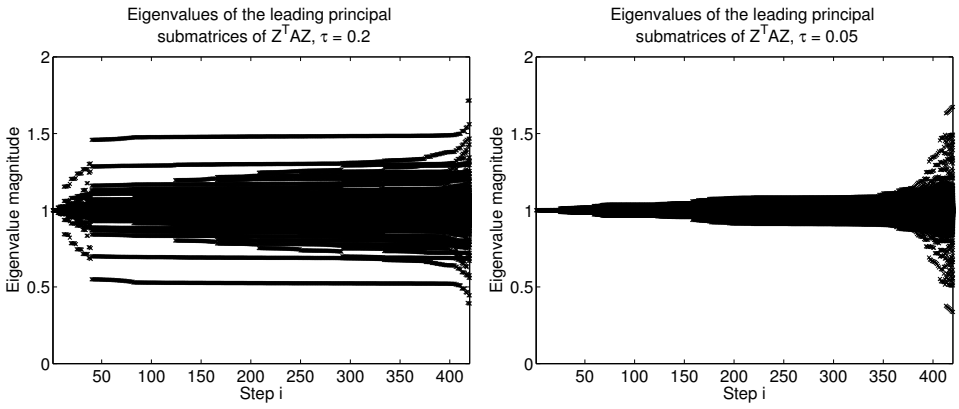

Figure 7: Figures showing eigenvalues of the leading principal submatrices of the matrix $\tilde{Z}^T A \tilde{Z}$ providing similar iteration counts. The results for the basic a posteriori filtering without reordering *(iters = 20, nnz($\tilde{Z}$) = 60659)* is depicted on the left, the results for the a posteriori with dynamic pivoting *(iters = 23, nnz($\tilde{Z}$) =19827)* are on the right.

6 Conclusions

This paper gives a new insight into the behavior of the generalized Gram–Schmidt based preconditioning. In particular, it proposes a new dropping strategy that seems to provide preconditioners that are sparse and powerful at the same time. The strategy is theoretically motivated and it naturally introduces adaptivity into the dropping. The theoretical insight is accompanied by some experimental results for a matrix arising in structural mechanics. Although we consider here only one matrix, the experiments represent a wider class of the real-world problems. Our future research will include the extension to the standard incomplete Cholesky decomposition by exploiting its tight connection with the considered approximate inverse decomposition.

Acknowledgement

This work was supported by Grant Agency the Czech Republic under the project 108/11/0853 and the project P201/13-06684S of the Grant Agency of the Czech Republic.

References

[1] T. Dupont, R.P. Kendall, H.H.J. Rachford, "An Approximate Factorization Procedure for the Solving Self-Adjoint Elliptic Difference Equations", *SIAM J. Numer. Anal.*, 5: 559–573, 1968.

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

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

[4] M. Benzi, C.D. Meyer, M. Tůma, "A sparse approximate inverse preconditioner for the conjugate gradient method", *SIAM J. Sci. Comput.*, 17(5): 1135–1149, 1996.

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

[6] M. Benzi, "Preconditioning techniques for large linear systems: a survey", *J. Comput. Phys.*, 182(2): 418–477, 2002.

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

[8] 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, URL `http://dx.doi.org/10.4203/ccp.101.21`, paper 21.

[9] G.H. Golub, C.F. Van Loan, *Matrix Computations. 3rd ed.*, The Johns Hopkins University Press, Baltimore and London, 1996.

[10] S. Bellavia, J. Gondzio, B. Morini, "A Matrix-Free Preconditioner for Sparse Symmetric Positive Definite Systems and Least-Squares Problems", *SIAM J. Sci. Comput.*, 35(1): 192–211, 2013.