Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015.
Trabalho apresentado no XXXV CNMAC, Natal-RN, 2014.

# An Overview of Approximate Inverse Methods

**Luiz M. P. Carvalho**, **João P. Zanardi**, **Ítalo Nievinski**
UERJ - FEN, IME
20.550-900, Rio de Janeiro, RJ
E-mail: lmz@uerj.br, {jpzanardi,italovienski}@gmail.com,

**Michael Souza**
UFC - DEMA
60455-760, Fortaleza, CE
E-mail: michael@ufc.br

**Abstract:** *A comparison between sparse approximate inverse preconditioner is given. A simplified description of the considered preconditioners is presented, and the results of numerical experiments with academic examples are discussed.*

**Keywords:** *preconditioner, approximate inverse, sparse matrix*

## 1 Introduction

In this paper we describe and compare some sparse inverse approximate preconditioners for general matrices. We focus on Krylov subspace methods based on approximate inverse preconditioning (AINVP) for the iterative solution of $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is supposed to be nonsingular and $x, b \in \mathbb{R}^n$.

In this work approximate inverse preconditioners [4, 19] are matrices that approximate the inverse of the original matrix $A$, and the new system reads

$$MAx = Mb, \quad M \approx A^{-1}. \tag{1}$$

Equation (1) describes the action of a left preconditioner. The action of a right preconditioner can be formulated in a similar way as $AMy = b$, with $x = My$.

The preconditioners based on incomplete $LU$ factorizations have been losing their importance due to their intrinsic difficulty to run on parallel machines [17]. AINVP methods are highly parallelizable as they are applied through a simple matrix-vector multiplication, so they can become natural substitutes of the ILU alternatives; however, even if $A$ is sparse its inverse can be dense, which is a major concern. Consequently, our goal is to study approximate inverse preconditioners that have a fair compromise between parallelization and memory usage. Another compelling point is the parallel construction of this class of preconditioners.

This article is a computational comparison of some preconditioners alternatives based on approximations of inverse [3, 4, 7, 8, 9, 10, 13, 14]. We have implemented sequential codes, in Matlab and/or C and tested these codes by solving some academic problems.

This report is organized as follows. In section 2 we discuss the general background of this study through the analysis of three families of approximate inverse preconditioners. Section 3 has numerical experiments and discussions about them. Finally we draw some conclusions and future work in section 4.

DOI: 10.5540/03.2015.003.01.0097

## 2 Overview of approximate inverse methods

Following [4] we consider two categories of preconditioners: (a) approximate inverse methods based on Frobenius norm minimization, and (b) factorized approximate inverses. For a broader overview of these methods, we suggest [4] and [19, p. 298].

### 2.1 Methods based on Frobenius norm minimization

The basic idea of this class of approximate inverse techniques is to compute the preconditioner $M \approx A^{-1}$ as the solution of the following problem:

$$\min_{M \in S} \|I - AM\|_F^2,$$

where $S$ is a set of sparse matrices and $\|\cdot\|_F$ denotes the Frobenius norm. Note that

$$\|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_j\|_2^2,$$

where $e_j$ and $m_j$ denotes the $j-$th columns of the identity matrix and $M$, respectively. So the computation of $M$ is reduced to $n$ independent problems constrained to $S$, i.e.,

$$\min_{m_j} \|e_j - Am_j\|_F^2. \tag{2}$$

Given a sparsity pattern $G \subseteq \{(i,j) | 1 \leq i,j \leq n\}$, we define $S$ as the set of all real $n \times n$ matrices with nonzero pattern contained in $G$, i.e., $S = \{M \in \mathbb{R}^{n \times n} : m_{ij} = 0 \ \ \forall (i,j) \in G\}$. In this case, the nonzero entries in a column $m_j$ can be computed by solving a small least squares problem considering just the possible nonzero entries of $M$.

For general matrices it is hard to prescribe a good nonzero pattern. An alternative is to use an adaptive strategy that starts with a naive initial guess (e.g., a diagonal pattern) which is successively modified until $\|e_j - Am_j\| <$ `TOL` (where `TOL` is a given tolerance) or a maximum number of nonzeros in $m_j$ is reached.

One of the most successful approaches to the Frobenius norm formulation is the **SPAI** preconditioner proposed by Grote and Huckle [13]. The serial cost of computing and the storage requirements of **SPAI** preconditioner can be high. So, aiming to mitigate these problems, Chow and Saad proposed the Minimal Residual (**MR**) algorithm which uses a few steps of minimal residual-type method [9]. The sparsity of the preconditioner is guaranteed by dropping elements that are small in magnitudes or by using a given sparsity pattern.

### 2.2 Factorized sparse approximate inverses

We now consider preconditioners based on incomplete factorizations of $A^{-1}$. If $A = LDU$ is the $LDU$ decomposition of $A$, then $A^{-1} = U^{-1}D^{-1}L^{-1}$. The idea is to compute two sparse matrices $Z \approx U^{-1}$ and $W \approx L^{-T}$ and thus the factorized approximate inverse reads $M = ZD^{-1}W^T$.

In the **AINV** method proposed by Benzi and Tůma [1, 2, 3], the approximate inverse factors are calculated using a biconjugation scheme that generates two sets of vectors $\{z_i\}_{i=1}^n$, $\{w_i\}_{i=1}^n$, which are $A$-biconjugated. The factors $W = [w_1, \ldots, w_n]$ and $Z = [z_1, \ldots, z_n]$ can be rather dense, to alleviate this problem elements $z_{ij}$ and $w_{ij}$ that are smaller than a prescribed threshold are dropped.

Bru et al [6, 7, 8] proposed an alternative algorithm called **NBIF** to calculate the factors $W$ and $Z$ based on the Inverse of Sherman-Morrison (**ISM**)

$$(A + xy^t)^{-1} = A^{-1} + \frac{A^{-1}xy^tA^{-1}}{1 + y^tA^{-1}x}.$$

 

One of interesting features of **NBIF** is that it gives an approximation of the $LDU$ factorization and also an approximation of the inverse factors $L^{-1}$ and $U^{-1}$. The drop strategy of the **NBIF** algorithm uses the incomplete approximations of both factorizations. Let $w_{ik} = W_{ik}$ be the element of the $i-$th row and $k-$th column of the approximate factor $W \approx L^{-1}$ and $\epsilon$ a drop tolerance, then $w_{ik}$ is dropped if $|w_{ik}|\|e_k^T \bar{L}\| \leq \epsilon$, where $\bar{L} \approx L$. This coupled drop strategy was developed by Bollhöfer and Saad [5].

Kolotilina and Yeremim proposed a variation of **SPAI** known as **FSAI** to compute the inverse factors approximation [15, 16]. The idea is define two sets of indices $G_L$ and $G_U$ such that

$$\{(i,j)|i<j\} \subseteq G_L \subseteq \{(i,j)|i \neq j\} \text{ and } \{(i,j)|i>j\} \subseteq G_U \subseteq \{(i,j)|i \neq j\},$$

which give the sparsity pattern (zero entries) of $W \approx L^{-1}$ and $Z \approx U^{-1}$. Then factors entries are calculated using the equations

$$\begin{aligned} w_{ij} &= 0 \ (i,j) \in G_L \\ z_{ij} &= 0 \ (i,j) \in G_U \\ (WA)_{ij} &= \delta_{ij} \ (i,j) \notin G_L \\ (AZ)_{ij} &= \delta_{ij} \ (i,j) \notin G_U. \end{aligned}$$

Using these equations, each column of the approximate inverse factors can be calculated from reduced linear systems following ideas similar to the ones of **SPAI** method. The preconditioned matrix can be written as $WAZD^{-1}$, where $D = diag(WAZ)$. The approximate factor $W$ computed by **FSAI** method minimizes $||I - ZWLU||_F$ where $L, U$ are the LU factors of $A$, subject to the sparsity patterns $G_L, G_U$ [16].

## 3 Numerical experiments

In this section we present the numerical performance of the preconditioners previously described. We implemented the methods `SPAI-S`, `SPAI-A`, `AINV` and `NBIF` in C language and `MR-A`, `MR-S` and `FSAI` in Matlab language. The methods `SPAI-S` and `SPAI-A` are implementations of **SPAI** method using respectively static and dynamic sparsity parameters, and `MR-A`, `MR-S` are their variations based on the **MR** approach. We also analyzed the performance of two versions of the classical incomplete LU factorization preconditioner. The first one controlled by level of fill-in, `ILU(level)`, and the second based on a dropping strategy, `ILUT`, both are available in the Matlab environment, but implemented in FORTRAN [19].

We tested `ILU(0)` and `ILUT(`$1E-2$`)`. All the computations were done in double precision on a Dell XPS 14 with CPU Intel Core i5-3337U 1.8GHz and 8GB of memory, the compiler environment was GNU C/C++ 4.6.3 in Ubuntu Linux.

The experiments that we present regard the solution of a linear system $Ax = b$ using preconditioned `GMRES` with restart after 30 iterations and initial guess $x^t = (0, \ldots, 0)$. We compare the number of iterations demanded by `GMRES` using each of the preconditioners listed above and also without any preconditioner. We also report the ratio $nnz(M)/nnz(A)$ between the number of nonzeros of the preconditioner and the number of nonzero elements in the coefficient matrix $A$. The maximum number of nonzeros in the preconditioners was around twice the number of nonzeros in the coefficient matrix $A$.

The first experiment is based on a matrix associated with the modeling of three-dimensional single-phase incompressible flow in porous medium with a seven-point block-centered finite-difference discretization of the differential operator $\nabla \cdot K\nabla$, imposing Dirichlet boundary conditions. We generated a $12 \times 12$ "tile" of values for $K$, as shown in the Figure 1. This tile is a realization of a stationary log-normal distribution with an anisotropic Gaussian covariance function (see [18], Section 5.4). Each layer of the field $K$ was then created by the tiling of this $12 \times 12$ module (upon suitable reflections, in order to maintain smoothness). All layers were

 

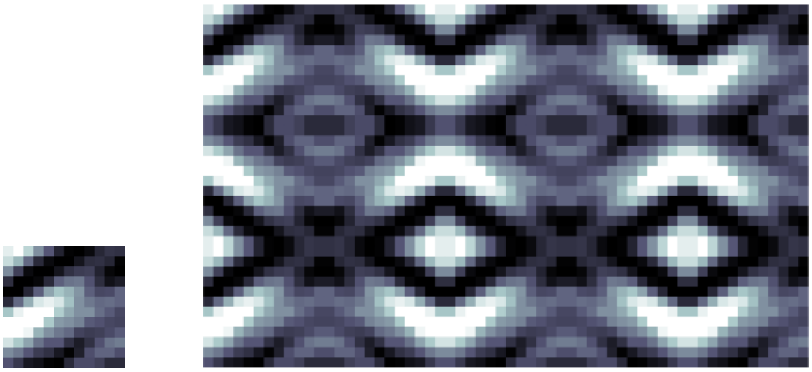

Figure 1: On the left, the basic $12 \times 12$ tile used in the experiments with heterogeneous field. On the right, a composition of $3 \times 5$ such tiles, suitably reflected.

made identical (i.e., $K = K(x, y)$ and not $K = K(x, y, z)$.) The resulting fields were heterogeneous (with values varying two orders of magnitude), yet isotropic. The tile and the resulting $5 \times 3$ tiling are shown in Figure 1. The second experiment is based on the discretization of the operator $L(u) = \nabla \cdot (u\vec{v}) - \epsilon \Delta u$, related to the convection-diffusion equation.

Table 1 gives, for each matrix $A$, the order $n$ and the number of nonzeros $nnz(A)$. The right-hand side of each linear system was computed from the solution vector $x^*$ of all ones, the choice used, e.g., in [3, 20]. In all experiments the relative residual $|b - Ax|/|b|$ was less than 1E-07.

| ALL INSTANCES | $n$ | $nnz(A)$ |
|---|---|---|
| $12 \times 12 \times 12 = 12^3$ | 1728 | 11232 |
| $36 \times 36 \times 36 = 36^3$ | 46656 | 318816 |
| $48 \times 48 \times 48 = 48^3$ | 110592 | 760320 |

Table 1: The size of matrix $A$ and its number of nonzero elements for all instances.

In Table 2, we show test results for instances Heterogeneous and Convection-Diffusion. The most significant improvement in the `GMRES` iterations among the approximate inverse preconditioners were obtained by the methods `AINV` and `NBIF`. The `AINV` algorithm showed a performance

| | **Heterogeneous** | | | | | | **Convection-Diffusion** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **iGMRES** | | | **nnz(M)/nnz(A)** | | | **iGMRES** | | | **nnz(M)/nnz(A)** | | |
| **Method** | $12^3$ | $36^3$ | $48^3$ | $12^3$ | $36^3$ | $48^3$ | $12^3$ | $36^3$ | $48^3$ | $12^3$ | $36^3$ | $48^3$ |
| `GMRES` | 162 | $> 10^3$ | $> 10^3$ | 0.00 | 0.00 | 0.00 | 120 | 345 | 419 | 0.00 | 0.00 | 0.00 |
| `ILU(0)` | 12 | 53 | 70 | 1.00 | 1.00 | 1.00 | 11 | 25 | 31 | 1.00 | 1.00 | 1.00 |
| `ILUT` | 6 | 18 | 22 | 1.46 | 1.56 | 1.65 | 3 | 6 | 7 | 1.62 | 1.78 | 1.80 |
| `SPAI-S` | 24 | 131 | 179 | 1.00 | 1.00 | 1.00 | 23 | 124 | 185 | 1.00 | 1.00 | 1.00 |
| `SPAI-A` | 24 | 123 | 124 | 1.13 | 1.13 | 0.87 | 20 | 68 | 126 | 1.45 | 1.53 | 1.54 |
| `FSAI` | 23 | 127 | 147 | 1.00 | 1.00 | 1.00 | 17 | 88 | 156 | 1.00 | 1.00 | 1.00 |
| `MR-S` | 29 | 168 | 225 | 1.00 | 1.00 | 1.00 | 26 | 123 | 185 | 1.00 | 1.00 | 1.00 |
| `MR-A` | 79 | $> 10^3$ | $10^3$ | 1.69 | 1.61 | 1.60 | 28 | 156 | 235 | 1.69 | 1.61 | 1.60 |
| `AINV` | 17 | 102 | 91 | 1.42 | 1.44 | 1.45 | 13 | 38 | 60 | 1.42 | 1.43 | 1.43 |
| `NBIF` | 17 | 60 | 76 | 1.68 | 2.07 | 2.19 | 12 | 31 | 52 | 1.48 | 1.54 | 1.55 |

Table 2: Results of instances Heterogeneous and Convection-Diffusion with $\epsilon = 1.0E - 5$.

DOI: 10.5540/03.2015.003.01.0097

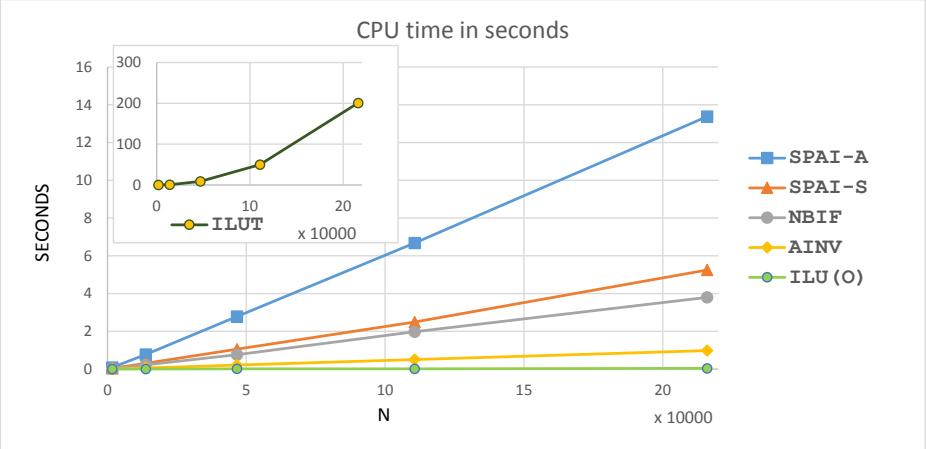


Figure 2: CPU time in seconds in Heterogeneous instance with $\epsilon = 1.0E-05$.

similar to NBIF but with a smaller number of nonzero elements (sparser preconditioner). The best improvement overall was obtained by the classical method ILUT, but its performance should be balanced by the fact that both ILUT and ILU(0) are essentially sequential in the construction and application phases which is an undesirable feature in the modern scenario of parallel heterogeneous architectures.

In our last experiment, we compare the CPU time between the sparse approximate inverse preconditioners SPAI-A, SPAI-S, AINV and NBIF coded in C language with the ILU(0) and ILUT. The other approximate inverse methods implemented in Matlab language were not considered in this experiment due the intrinsic poor performance of the Matlab when dealing with sparse matrix accessed through its indexes. Figure 2 shows the results. The ILU(0) was extremely fast while ILUT was the slowest. Despite the calculations of direct and inverse factors, the NBIF algorithm also presents a very good performance been the third fastest. All preconditioners, except ILUT, showed linear complexity with respect to the problem size given by the number of columns in the coefficient matrix $A$. It is important to note that the quadratic behavior of ILUT may hinder its usage in applications with large linear systems.

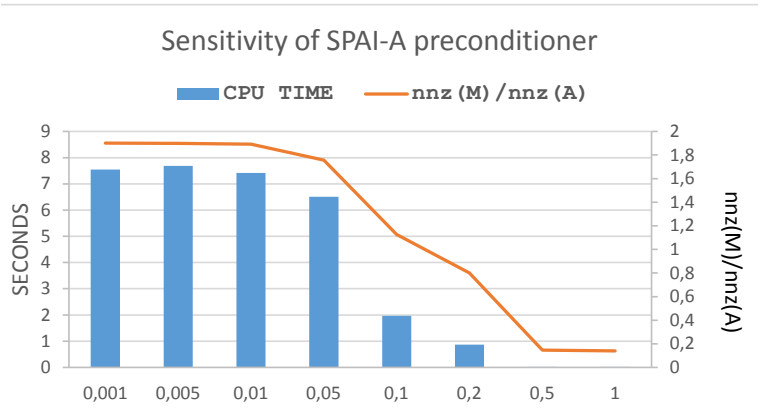


Figure 3: CPU time in seconds and the ration $nnz(M)/nnz(A)$ between the number of nonzeros of preconditioner and the number of nonzeros in the coefficient matrix $A$ in the Heterogeneous instance with $\epsilon = 1.0E-05$.

At a first glance the results may suggest that some of the approximate inverse algorithms could be excluded from future consideration, as their performance were not so good as the best

 

methods. However all of the approximate inverse methods have many parameters that could change their performance. In fact, Figure 3 shows the sensitivity of `SPAI-A` with respect to the parameter `TOL` associated with the stopping criterion. As we can see, the sparsity and the consequent efficiency of the `SPAI-A` algorithm are highly sensitive to this parameter. This sensitivity to parameters is shared by all approximate inverse methods that we have tested. So it would be premature to conclude a clear superiority of any of the presented methods based solely on the results of these numerical experiments.

## 4 Future work

We have compared the performance of several approximate inverse preconditioners for solving academic problems. We also compared the performance of approximate inverse preconditioners with standard methods `ILU(0)` and `ILUT`. Our results indicate that the factorized versions **NBIF** and **AINV** are the most effective among the approximate inverse preconditioners. Considering the complexity with respect to the number of columns of the coefficient matrix $A$, `ILUT` showed a quadratic behavior while all other methods behaved linearly. The amount of CPU time demanded by `ILUT` was more than one hundred times bigger than the one required by **NBIF** and **AINV** for instance.

Due to the sensitivity to the parameters and the small number of instances, it would be premature to conclude a definitive superiority of any of the presented preconditioners. In order to support our conclusions we plan to develop an automatic procedure to identify good values for these parameters using a tabu search algorithm to explore the large space of parameters [11, 12]. We also plan to test these preconditioners for solving matrices from oil reservoir simulations.

As we target parallel hybrid machines, our next step is to implement parallel versions of some of these preconditioners using PETSc. In a second moment, we will implement the main linear algebra kernels of this preconditioners in accelerators as NVIDIA's GPU and Intel's Xeon Phy.

## References

[1] M. Benzi. *A direct row-projection method for sparse linear systems.* PhD thesis, North Carolina State University, 1993.

[2] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

[3] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 19(3):968–994, 1998.

[4] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2):305–340, 1999.

[5] M. Bollhöfer and Y. Saad. On the relations between ilus and factored approximate inverses. *SIAM Journal on Matrix Analysis and Applications*, 24(1):219–237, 2002.

[6] R. Bru, J. Cerdán, J. Marín, and J. Mas. Preconditioning sparse nonsymmetric linear systems with the Sherman–Morrison formula. *SIAM Journal on Scientific Computing*, 25(2):701–715, 2003.

[7] R. Bru, J. Marín, J. Mas, and M. Tůma. Balanced incomplete factorization. *SIAM Journal on Scientific Computing*, 30(5):2302–2318, 2008.

 

[8] R. Bru, J. Marín, J. Mas, and M. Tůma. Improved balanced incomplete factorization. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2431–2452, 2010.

[9] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, 19(3):995–1023, 1998.

[10] J. Cosgrove, J. Diaz, and A. Griewank. Approximate inverse preconditionings for sparse linear systems. *International journal of computer mathematics*, 44(1-4):91–110, 1992.

[11] F. Glover. Tabu search, part I. *ORSA Journal on computing*, 1(3):190–206, 1989.

[12] F. Glover. Tabu search, part II. *ORSA Journal on computing*, 2(1):4–32, 1990.

[13] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[14] T. Huckle and A. Kallischko. Frobenius norm minimization and probing for preconditioning. *International Journal of Computer Mathematics*, 84(8):1225–1248, 2007.

[15] L. Y. Kolotilina, N. AA, and Y. A. Yu. Factorized sparse approximate inverse preconditioning. iv:simple approaches to rising efficiency. *Numerical Linear Algebra with Applications*, 6:515–531, 1999.

[16] L. Y. Kolotilina and Y. A. Yu. Factorized sparse approximate inverse preconditioning. i:theory. *SIAM Journal on Matrix Analysis and Applications*, 14:45–58, 1993.

[17] R. Li and Y. Saad. Divide and conquer low-rank preconditioners for symmetric matrices. *SIAM Journal on Scientific Computing*, 35(4):A2069–A2095, 2013.

[18] D. Oliver, A. Reynolds, and N. Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambridge University Press, 2008.

[19] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

[20] Z. Zlatev. *Computational methods for general sparse matrices*, volume 65. Springer, 1991.

 