Structured Iterative Approximations in Numerical Linear Algebra

JuliaCon 2026
Johannes Gutenberg University

Nicolas Venkovic and Hartwig Anzt

Chair of Computational Mathematics
School of Computation, Information and Technology
Technical University of Munich

August 13, 2026

Outline I

1. Introduction 1
2. Sparse Approximate Inverses (SPAIs) 2
3. Approximate low-rank CP tensor representation 15

Introduction

The "What?", the "Why?" and the "How?"

- A few *approximation problems* of interest to us:
  - **Linear solves** with $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$:

$$\text{Find } x \in \mathbb{R}^n \text{ s.t. } Ax \approx b.$$

  - **Inversion** of $A \in \mathbb{R}^{n \times n}$:

$$\text{Find } M \in \mathbb{R}^{n \times n} \text{ s.t. } M \approx A^{-1}.$$

  - **Low-rank CP representation** of $\mathfrak{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$:

$$\text{Find } (U_1, \ldots, U_d) \in \mathbb{R}^{n_1 \times r} \times \cdots \times \mathbb{R}^{n_d \times r} \text{ s.t. } [\![U_1, \ldots, U_d]\!] \approx \mathfrak{X}$$

    for some $r \ll \min\{n_1, \ldots, n_d\}$ where

$$[\![U_1, \ldots, U_d]\!] := \sum_{k=1}^{r} \underbrace{U_1[:, k]}_{u_k^{(1)} \in \mathbb{R}^{n_1}} \circ \cdots \circ \underbrace{U_d[:, k]}_{u_k^{(d)} \in \mathbb{R}^{n_d}} .$$

- Interests in both:
  - unconstrained variants of these problems, and the case of
  - **structured** (e.g., sparse, nonnegative, symmetric, definite, ...) **iterates**.

Sparse Approximate Inverses (SPAIs)

Sparse approximate inverses

- Numerical linear algebraic (NLA) problem of interest:
  - Finding an approximate inverse of a matrix $A \in \mathbb{R}^{n\times n}$:

$$\boxed{M \in \mathbb{R}^{n\times n} \text{ s.t. } AM \approx I_n}$$

  such that $x \mapsto Mx$ can computed efficiently irrespective of x.
- Approximate inverses M, or at least, maps of the form $x \mapsto Mx$, are routinely used in NLA to accelerate the convergence of iterative solvers. For example,

$$\begin{aligned} \text{Instead of solving for } x \text{ s.t. } Ax = b, \\ \text{we solve for } u \text{ s.t. } AMu = b. \\ \text{Then, we have } x = Mu. \end{aligned}$$

- When A is sparse (i.e., most components are zero), although A^{-1} is generally dense (i.e., not sparse), sparse approximate inverse (SPAI) matrices M can nevertheless be good candidates to accelerate iterative computations by preconditioning.

Best SPAIs

- For prescribed sparsity patterns (i.e., for a given set of nonzero components in M), we can leverage the fact that

$$\|I_n - AM\|_F^2 = \sum_{i=1}^{n} \|e_i^{(n)} - Am_i\|_2 \text{ where } m_i := Me_i^{(n)}$$

(the i-th column of M)

to find the "best" SPAI M through decoupled sparse least squares solves.

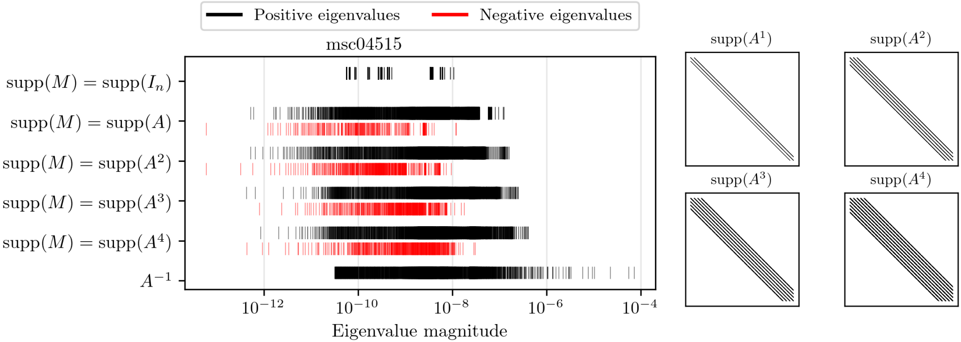
- We generally do not know what the sparsity pattern of M should be to efficiently accelerate iterative computations with A.

  Standard choices are nonzero patterns based on monomials of A, i.e., sparsity patterns of A, of A^2, of A^3, ...

  But those choices sometimes fail to yield good SPAIs.

Limits of best SPAIs with sparsity of low powers of A

- Example of matrix for which the best SPAI with sparsity patterns of $A, A^2, A^3, \ldots$ fail to reproduce the definiteness of A^{-1}:



Alternative strategies exist that iterate from one sparsity pattern to another, gradually building a better SPAI.

We call those approaches sparse matrix iteration.

State-of-the-art sparse matrix iteration

- Starting with an initial SPAI M_0, sparse matrix iterations build sequences of SPAI iterates $M_1, M_2, \ldots$ by trying to minimize

$$f(M) := \|I_n - AM\|_F^2 \text{ with } \nabla_M f(M) \propto A^T R$$

  where $R := I_n - AM$ denotes the residual.
- The Newton-Schulz iteration is obtained by considering scalar inversion through Newton-Raphson iteration and replacing the scalar iterate by an SPAI. This leads to:

$$M_{i+1} := M_i(3I_n - M_i^2)/2.$$

- Steepest descent (SD) iterations are defined by the recurrence formula

$$M_{i+1} := \arg \min_{M \in M_i + \mathsf{span}\{A^T R_i\}} f(M)$$

  which is equivalently defined by the following projection:

$$M_{i+1} \in M_i + \mathsf{span}\{A^T R_i\} \text{ s.t. } R_{i+1} := I_n - AM_{i+1} \perp A\,\mathsf{span}\{A^T R_i\}.$$

State-of-the-art sparse matrix iteration, cont'd

- Faster convergence than Newton-Schulz and SD is achieved by minimal residual (MR) iteration:

$$M_{i+1} := \arg \min_{M \in M_i + \mathsf{span}\{R_i\}} f(M)$$

which is achieved by the following projection:

$$M_{i+1} \in M_i + \mathsf{span}\{R_i\} \text{ s.t. } R_{i+1} := I_n - AM_{i+1} \perp A\, \mathsf{span}\{R_i\}.$$

- Sparse matrix iterations are conducted using sparse data structures with sparse basic linear algebraic kernels.

  As a result, the sparsity pattern evolves naturally from M_i to M_{i+1}.
- Even "learning" sparsity through such sparse matrix iterations can sometimes fail to produce good SPAIs.

Chow, E., & Saad, Y. (1998). Approximate inverse preconditioners via sparse-sparse iterations. SIAM Journal on Scientific Computing, 19(3), 995-1023.

Other variants of sparse matrix iteration

- Venkovic and Anzt (2025) introduce LOMR iterations as a locally enriched variant of MR:

$$M_{i+1} := \arg \min_{M \in M_i + \text{span}\{P_{i-1}, R_i\}} \|I_n - AM\|_F \;\text{ for }\; i = 0, 1, \dots .$$

with (sparse) search directions $P_0, P_1, \dots$ updated by:

$$M_{i+1} := M_i + \underbrace{\delta_i R_i + \gamma_i P_{i-1}}_{P_i} \;\text{ for }\; i = 0, 1, \dots$$

- In that work, we also advocate for the computation of SPAIs using sparse matrix conjugate gradient (CG) iterations:

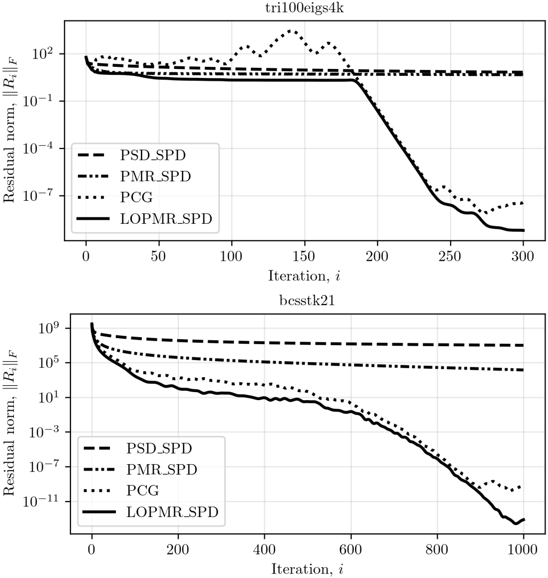
$$M_i \in M_0 + \mathcal{K}_i(A, M_0) \text{ s.t. } R_i := I_n - AM_i \perp \mathcal{K}_i(A, M_0)$$

where $\mathcal{K}_i(A, M_0) := \text{span}\{M_0, AM_0, \dots, A^{i-1}M_0\}$ is a Krylov subspace.

N. Venkovic and H. Anzt (2025) Global iterative methods for sparse approximate inverses of symmetric positive-definite matrices. arXiv preprint, arXiv:2511.09753.

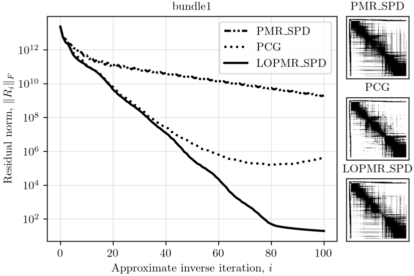
Comparison of matrix iterations

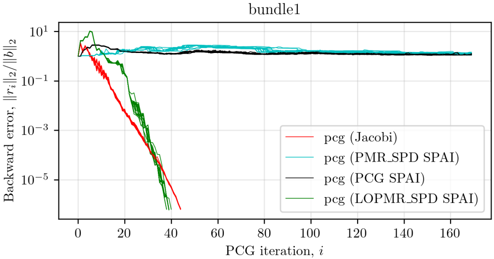
- LOMR typically acts as a lower envelope to matrix CG iterations:



Results with nonzero dropping

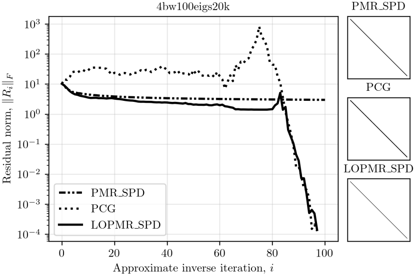
- For some matrices, SPAIs obtained by matrix iteration fail to consistently outperform Jacobi preconditioners:

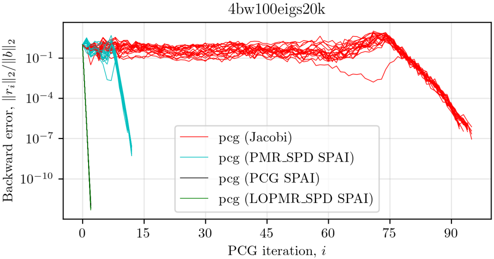




Results with nonzero dropping, cont'd

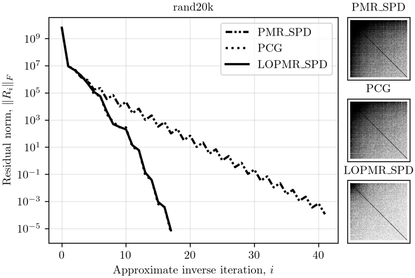
- For some matrices, SPAIs obtained by matrix iteration fail to consistently outperform Jacobi preconditioners:

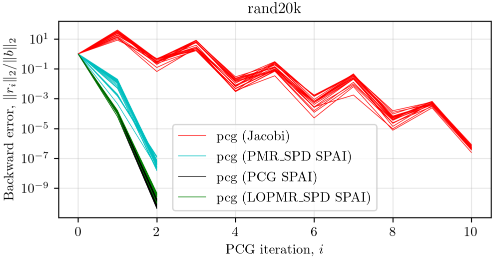




Results with nonzero dropping, cont'd

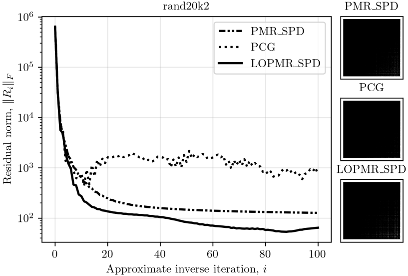
- For some matrices, all three matrix iterations produce equally good SPAIs:

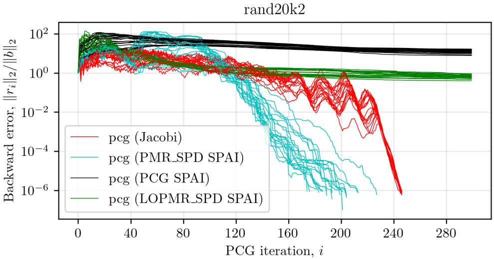




Results with nonzero dropping, cont'd

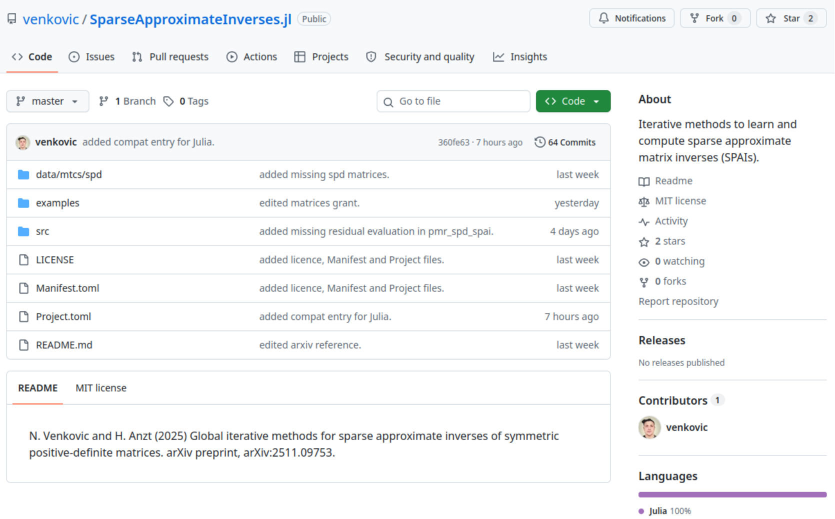
- For some matrices, the dropping of nonzero components plays a crucial role in finding a good SPAI:





SparseApproximateInverses.jl

- All these methods are made available on this repository:



What else is going on?

- The computational cost of every matrix iteration is significant:

Method	Operation count per iteration
MR	2 SpGEMM + 1 SpFIP + 1 SpFN + 2 SpGEAMs
SD	3 SpGEMMs + 1 SpFIP + 1 SpFN + 2 SpGEAMs
CG	2 SpGEMMs + 1 SpFIP + 1 SpFN + 3 SpGEAMs
LOMR	3 SpGEMMs + 3 SpFIPs + 2 SpFNs + 4 SpGEAMs

†: General product of sparse matrices $^\circ$: Frobenius inner product of sparse matrices

$^\bullet$: Frobenius norm of sparse matrix *: General addition of sparse matrices and we

are currently investigating ways to alleviate those costs.

Mixed precision:

- Do as many costly parts of matrix iterations as possible in lower precision arithmetic.
- Coupling the learning of the sparsity pattern in low precision with best-SPAI computation for said pattern works in some cases.

Self-preconditioned variants:

- Outer-inner iteration parallelizable variant where the computed SPAI is used as preconditioner to improve its quality.

Approximate low-rank CP tensor representation

Low-rank CP tensor approximation

- To construct low-rank CP tensor approximations of the d-way array $\mathfrak{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, we make use of the gradients

$$\nabla_{U_\ell} g : \mathbb{R}^{n_1 \times r} \times \cdots \times \mathbb{R}^{n_d \times r} \to \mathbb{R}^{n_\ell \times r} \text{ for } \ell = 1, \ldots, d$$

  of the non-convex objective function

$$g : (U_1, \ldots, U_d) \mapsto \|\mathfrak{X} - [\![U_1, \ldots, U_d]\!]\|^2.$$

- A Petrov-Galerkin gradient descent is formulated with approximates

$$U_{\ell,i+1} \in U_{\ell,i} + \mathsf{span}\left\{\nabla_{U_\ell} g(U_{1,i}, \ldots, U_{d,i})\right\}$$

  where $r \cdot n_\ell$ orthogonality constraints need be applied to close the system for $\ell = 1, \ldots, d$.
  For matrices ($d = 2$), we proceed with the following constraints

$$R_{i+1} V_i \perp \mathsf{span}\{R_i V_i\}$$
$$R_{i+1}^T U_i \perp \mathsf{span}\{R_i^T U_i\}$$

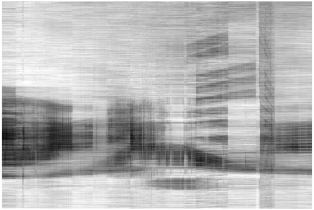
  which generalize to d-way arrays.

Comparison of methods

- Similarly as for approximate inverses, the Petrov-Galerkin gradient descent can be made locally optimal for faster convergence.

  Sample low-rank approximations achieved in fixed time (5 seconds):

  Petrov-Galerkin gradient descent

  Locally optimal variant

  Currently investigating applications of locally optimal variants for structured low-rank approximation.

  E.g., recovering sparse low-rank factors.