

Scalable Semi-Matrix-Free Preconditioning for Newton-Krylov Solvers

Mia Ohlrogge

Friedrich-Schiller-University Jena

28th EuroAD Workshop

December 9, 2025

Table of Contents

1 Model Problem and Numerical Approach

2 Numerical Experiments

3 Discussion and Conclusion

Model Problem: Two-Phase Flow

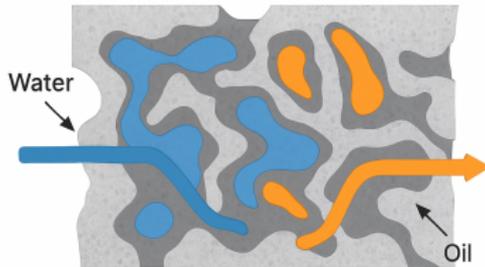


Figure: Two-phase flow in porous media

AI-generated picture

What is Two-Phase Flow?

- Simultaneous flow of two immiscible fluids – here: *water* and *oil*
- Governed by:
 - Mass conservation law
 - Darcy's law for velocity

⇒ System of **coupled, nonlinear PDEs**

⇒ **Numerically challenging problem**

Solving the Nonlinear System Numerically

Nonlinear system after discretization:

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{x} := (\mathbf{p}, \mathbf{S}_w)^T \in \mathbb{R}^{2N}$$

Solving $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ using **Newton's method** requires solving a linear system with the Jacobian J in every step:

$$J = \begin{bmatrix} \frac{\partial \mathbf{F}_w}{\partial \mathbf{p}} & \frac{\partial \mathbf{F}_w}{\partial \mathbf{S}_w} \\ \frac{\partial \mathbf{F}_o}{\partial \mathbf{p}} & \frac{\partial \mathbf{F}_o}{\partial \mathbf{S}_w} \end{bmatrix}$$

⇒ J is **large** and **sparse**

⇒ Idea: **GMRES** only needs **matrix-vector products**, not the full Jacobian

⇒ But: How do we apply **preconditioning** in a **matrix-free** setting?

Numerical Approach

Goal: *Build a preconditioner without assembling the full Jacobian.*

1. Determine the **sparsity pattern** of J and select **required entries** R
2. Solve a **graph coloring** problem to obtain a small set of **colors** p and construct the **seed matrix** $S \in \{0, 1\}^{n \times p}$
3. Compute a **compressed Jacobian** using **AD** with input S :

$$J_c = J \cdot S$$

4. Uncompress J_c to obtain a **sparse approximation** \tilde{J}
5. Apply **ILU**(\tilde{J}) to form the preconditioner M

Numerical Approach

Goal: *Build a preconditioner without assembling the full Jacobian.*

6. Solve with **matrix-free GMRES**:

- **AD** computes Jacobian-vector products Jv
-
- ✓ AD enables both preconditioning and matrix-free linear solves
 - ✓ Jv scales efficiently to large systems
 - ✓ The full Jacobian is never assembled – only a sparse approximation is used

Selection of Required Elements

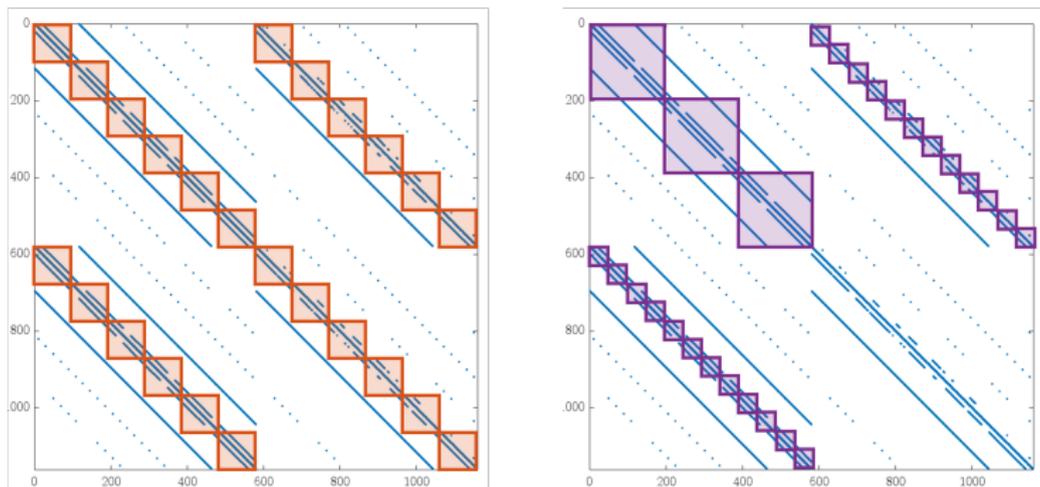


Figure: Different strategies for the selection of required elements using $k \times k$ blocks

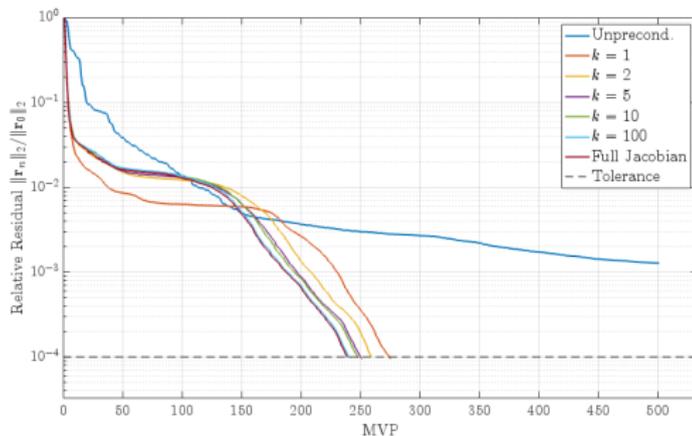
Table of Contents

1 Model Problem and Numerical Approach

2 Numerical Experiments

3 Discussion and Conclusion

GMRES Convergence



- All preconditioned systems **out-perform** the **unpreconditioned**
- **Increasing k improves convergence**, but gains diminish quickly

Figure: GMRES relative residual vs. the number of matrix-vector products for varying block sizes k .

Memory Requirement

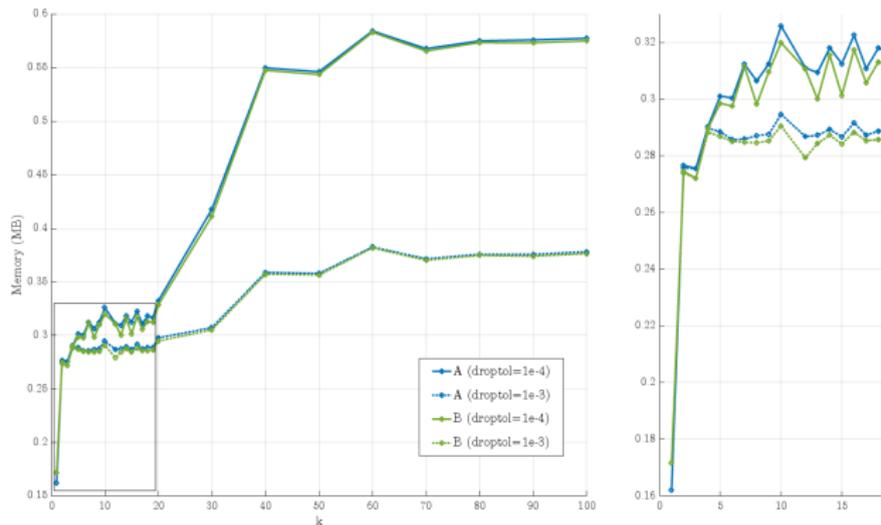


Figure: Memory requirement using varying preconditioners.

Solution Accuracy: Saturation

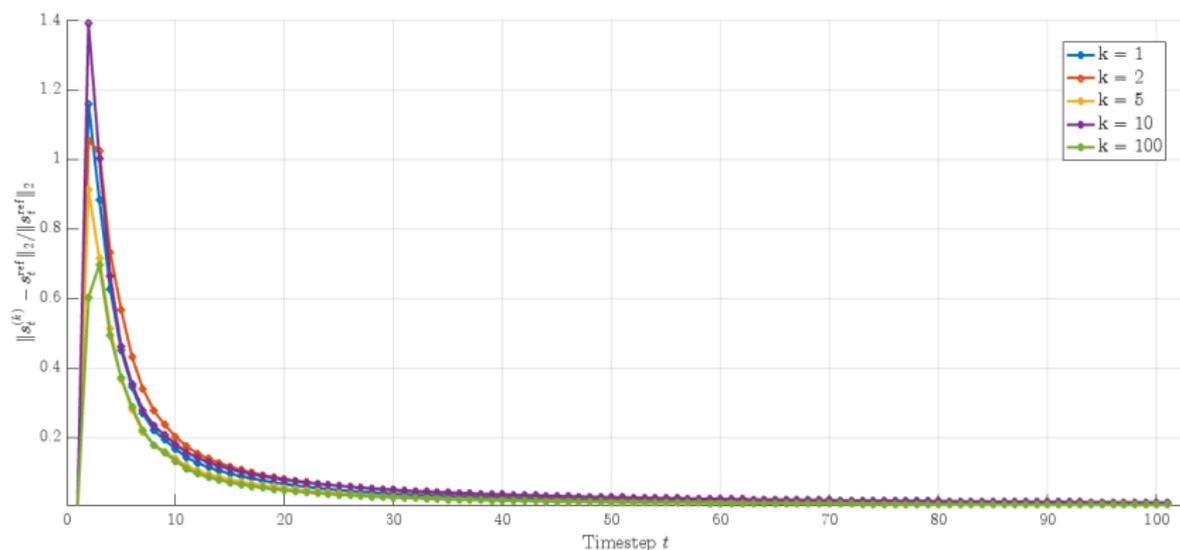


Figure: Relative saturation error during simulation for varying k .

Table of Contents

1 Model Problem and Numerical Approach

2 Numerical Experiments

3 Discussion and Conclusion

Discussion and Conclusion

Solver behavior and solution accuracy

- AD enables semi-matrix-free preconditioning
- The approach reduces memory without compromising stability
- No clear link between solver parameters and overall accuracy

Memory as the main constraint

- Memory usage grows with increasing block size k
- For large-scale problems, *minimizing memory* becomes critical

Outlook

- Dynamically choose *seed matrices* or *block sizes* during simulation
- Extend semi-matrix-free methods to other solvers and more complex PDEs