

Scalability Results for the Solution of the Richards Equation

F. Durastante

GIMC SIMAI Young 2022

Thursday, September 29th

MS 01 - Efficient linear solvers for coupled geophysical simulations.

Università di Pisa, ✉ fabio.durastante@unipi.it

Istituto per le Applicazioni del Calcolo “M. Picone” – CNR

Collaborators and Funding



Daniele Bertaccini

Università degli Studi di Roma
"Tor Vergata"
Dipartimento di Matematica
IAC-CNR



Pasqua D'Ambra,

Consiglio Nazionale delle Ricerche
Istituto per le Applicazioni del
Calcolo "M. Picone"



Salvatore Filippone,

Università degli Studi di Roma
"Tor Vergata"
Dipartimento di Ingegneria Civile
e Ingegneria Informatica
IAC-CNR



Horizon 2020
European Union funding
for Research & Innovation

The Richards Equation: a fluid flow model

Richards equation models fluid flow in the *unsaturated* (vadose) zone, it is

- ⚙️ **non-linear** the parameters that control the flow are dependent on the saturation of the media,
- ⚙️ a combination of **Darcy's law** and the principle of **mass conservation**

$$\frac{\partial (\rho \phi s(p))}{\partial t} + \nabla \cdot q = 0,$$

- ⚙️ $s(p)$ is the **saturation** at pressure head p of a fluid with density ρ and terrain porosity ϕ ,
- ⚙️ q is the volumetric water flux, using Darcy's law it is written as

$$q = -K(p) (\nabla p + c\hat{z}),$$

- ⚙️ $K(p)$ the **hydraulic conductivity**,
- ⚙️ c the cosine of the angle between the downward z-axis \hat{z} and the direction of the gravity force

The Richards Equation: constitutive equations

To close the model we need equations for both $s(p)$ and $K(p)$, we use the Van Genuchten formulation [Celia et al. 1990; Van Genuchten, 1980]

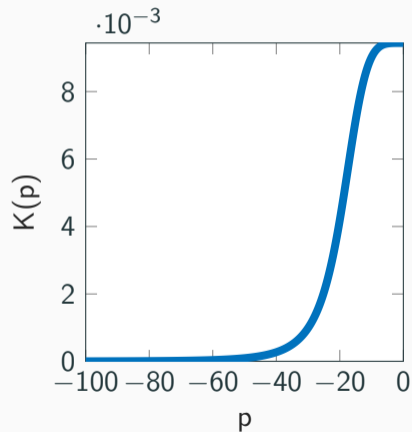
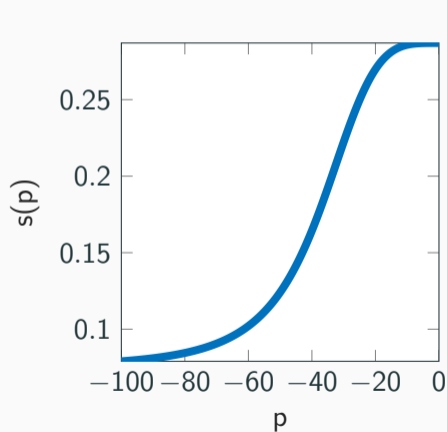
$$s(p) = \frac{\alpha(s_s - s_r)}{\alpha + |p|^\beta} + s_r, \text{ and } K(p) = K_s \frac{a}{a + |p|^\gamma},$$

where

- ⚙ all the parameters $(\alpha, \beta, \gamma, a)$ are **fitted on real data** and *assumed* to be *constant* in the media;
- ⚙ K_s is the saturated hydraulic conductivity.

The Richards Equation: constitutive equations

To close the model we need equations for both $s(p)$ and $K(p)$, we use the Van Genuchten formulation [Celia et al. 1990; Van Genuchten, 1980]



Cell-centered finite difference discretization

We use a discretization on a **cell-centered finite difference tensor mesh** on

- ⚙ a parallelepiped discretized with $\mathbf{N} = (N_x, N_y, N_z)$ nodes,
- ⚙ the cell centers $\{x_{i,j,k} = (ih_x, jh_y, kh_z)\}_{i,j,k=0}^{N-1}$, for
 $\mathbf{h} = (h_x, h_y, h_z) = (L_x, L_y, L_z)/(\mathbf{N} - 1)$;
- ⚙ the relative interfaces located at midpoints between adjacent nodes;
- ⚙ N_t uniform time steps, i.e., the grid $\{t_l = l\Delta t\}_{l=0}^{N_t-1}$ for $\Delta t = 1/(N_t - 1)$.

This gives the **non-linear equations**:

$$\Phi(p_{i,j,k}^{(l)}) = \frac{\rho\phi}{\Delta t} \left(s(p_{i,j,k}^{(l)}) - s(p_{i,j,k}^{(l-1)}) \right) + q_{i+1/2,j,k}^{(l)} - q_{i-1/2,j,k}^{(l)} + q_{i,j+1/2,k}^{(l)} - q_{i,j-1/2,k}^{(l)} \\ + q_{i,j,k+1/2}^{(l)} - q_{i,j,k-1/2}^{(l)} + f_{i,j,k} \equiv 0, \quad \text{for } i, j, k = 1, \dots, \mathbf{N} - 2,$$

Cell-centered finite difference discretization

$$\Phi(p_{i,j,k}^{(l)}) = \frac{\rho\phi}{\Delta t} \left(s(p_{i,j,k}^{(l)}) - s(p_{i,j,k}^{(l-1)}) \right) + q_{i+1/2,j,k}^{(l)} - q_{i-1/2,j,k}^{(l)} + q_{i,j+1/2,k}^{(l)} - q_{i,j-1/2,k}^{(l)} \\ + q_{i,j,k+1/2}^{(l)} - q_{i,j,k-1/2}^{(l)} + f_{i,j,k} \equiv 0, \quad \text{for } i, j, k = 1, \dots, \mathbf{N} - 2,$$

with

$$q_{i+1/2,j,k}^{(l)} = -{}^{\text{AV}}K_{i+1,i}^{(l)} \left(\frac{p_{i+1,j,k}^{(l)} - p_{i,j,k}^{(l)}}{h_x^2} \right), \quad q_{i-1/2,j,k}^{(l)} = -{}^{\text{AV}}K_{i-1,i}^{(l)} \left(\frac{p_{i,j,k}^{(l)} - p_{i-1,j,k}^{(l)}}{h_x^2} \right),$$

$$q_{i,j+1/2,k}^{(l)} = -{}^{\text{AV}}K_{j+1,j}^{(l)} \left(\frac{p_{i,j+1,k}^{(l)} - p_{i,j,k}^{(l)}}{h_y^2} \right), \quad q_{i,j-1/2,k}^{(l)} = -{}^{\text{AV}}K_{j-1,j}^{(l)} \left(\frac{p_{i,j,k}^{(l)} - p_{i,j-1,k}^{(l)}}{h_y^2} \right),$$

$$q_{i,j,k+1/2}^{(l)} = -{}^{\text{AV}}K_{k+1,k}^{(l)} \left(\frac{p_{i,j,k+1}^{(l)} - p_{i,j,k}^{(l)}}{h_z^2} \right) - \frac{K(p_{i,j,k+1})}{2h_z},$$

$$q_{i,j,k-1/2}^{(l)} = -{}^{\text{AV}}K_{k-1,k}^{(l)} \left(\frac{p_{i,j,k}^{(l)} - p_{i,j,k-1}^{(l)}}{h_z^2} \right) - \frac{K(p_{i,j,k-1})}{2h_z},$$

Values at the interfaces

The selection of the form of the average term that can lead to the more realistic simulations does **depend on the problem** and is **still an open problem**.

⚙ Denote by K_U and K_L the values of K on the opposite sides of the interface

arithmetic mean $^{ARIT}K = (K_U + K_L)/2,$

geometric mean $^{GEOM}K^{(I)} = \sqrt{K_U K_L},$

upstream-weighted mean

$$^{UP}K^{(I)} = \begin{cases} K_U, & p_U - p_L \geq 0, \\ K_L, & p_U - p_L < 0, \end{cases}$$

integral mean

$$^{INT}K^{(I)} = \begin{cases} \frac{1}{p_L - p_U} \int_{p_L}^{p_U} K(\psi) d\psi, & p_L \neq p_U, \\ K_U, & \text{otherwise.} \end{cases}$$

⚙ A combination of the above in the different directions

1. Estimate, fix all the parameters involved in the model, and select the opportune interface values for the discretization,

1. Estimate, fix all the parameters involved in the model, and select the opportune interface values for the discretization,
2. Select a time stepping method: Backward Euler with uniform steps,

1. Estimate, fix all the parameters involved in the model, and select the opportune interface values for the discretization,
2. Select a time stepping method: Backward Euler with uniform steps,
3. Iterative solution of the **nonlinear equation** with an inexact-Newton method

1. Estimate, fix all the parameters involved in the model, and select the opportune interface values for the discretization,
2. Select a time stepping method: Backward Euler with uniform steps,
3. Iterative solution of the **nonlinear equation** with an inexact-Newton method
4. **Iterative solution of the linear systems** with the Jacobian matrix:
 - ⚙ The Jacobian matrix $J = J_{\Phi}$ can then be computed in closed form,
 - ⚙ At the core of the (distributed) parallel solution resides the solution of the (right) preconditioned linear system

$$JM^{-1}(M\mathbf{d}_k) = -\Phi(\mathbf{p}^{(k,l)}),$$

Scalable AMG preconditioners

Algebraic MultiGrid methods are especially well suited for solving symmetric positive definite linear systems. Nevertheless, we are dealing with a **non symmetric** problem.

❓ What can we do?

🔧 Describe the **asymptotic spectral properties** of the sequence $\{J_N\}_N$,

Scalable AMG preconditioners

Algebraic MultiGrid methods are especially well suited for solving symmetric positive definite linear systems. Nevertheless, we are dealing with a **non symmetric** problem.

❓ What can we do?

- 🔧 Describe the **asymptotic spectral properties** of the sequence $\{J_N\}_N$,
- 🔧 Find an asymptotically *spectrally equivalent* symmetric positive definite matrix sequence $\{M_N\}_N$ to $\{J_N\}_N$,

Scalable AMG preconditioners

Algebraic MultiGrid methods are especially well suited for solving symmetric positive definite linear systems. Nevertheless, we are dealing with a **non symmetric** problem.

❓ What can we do?

- 🔧 Describe the **asymptotic spectral properties** of the sequence $\{J_N\}_N$,
- 🔧 Find an asymptotically *spectrally equivalent* symmetric positive definite matrix sequence $\{M_N\}_N$ to $\{J_N\}_N$,
- 🔧 Approximate $\{M_N\}_N$ by a **(parallel) AMG method** to efficiently solve the systems.

Scalable AMG preconditioners

Algebraic MultiGrid methods are especially well suited for solving symmetric positive definite linear systems. Nevertheless, we are dealing with a **non symmetric** problem.

❓ What can we do?

- 🔧 Describe the **asymptotic spectral properties** of the sequence $\{J_N\}_N$,
- 🔧 Find an asymptotically *spectrally equivalent* symmetric positive definite matrix sequence $\{M_N\}_N$ to $\{J_N\}_N$,
- 🔧 Approximate $\{M_N\}_N$ by a **(parallel) AMG method** to efficiently solve the systems.

As a **bonus** this will permit us to analyze the impact of (some) of the different **choices for the interface mean**.

The asymptotic spectrum of a sequence of matrices

To devise the preconditioners for these problems we want to **leverage on spectral information** about the sequence $\{J_N\}_N$

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N F(\lambda_i(J_N)) = \frac{1}{\mu_k(D)} \int_D F(f(\mathbf{x})) d\mathbf{x}, \quad \forall F \in C_c(\mathbb{C}),$$

- ⚙ f is a measurable function $f : D \subset \mathbb{R}^k \rightarrow \mathbb{C}$,
- ⚙ $\mu_k(\cdot)$ represent the Lebesgue measure on \mathbb{R}^k ,
- ⚙ $C_c(\mathbb{C})$ is the space of continuous functions with compact support.

Informal idea: “If we assume that N is large enough, then the eigenvalues of the matrix J_N , except possibly for $o(N)$ outliers, are approximately equal to the samples of f over a uniform grid in D ”

The expression in the Jacobian case

Theorem (Bertaccini, D'Ambra, D., Filippone)

The sequence $\{J_{\mathbf{N}}^{(k,j)}\}_{\mathbf{N}}$ obtained using either the **arithmetic** or **up-stream** averages, for $K(p)$, $s(p)$ given by the Van Genuchten model is distributed in the sense of the eigenvalues as the function

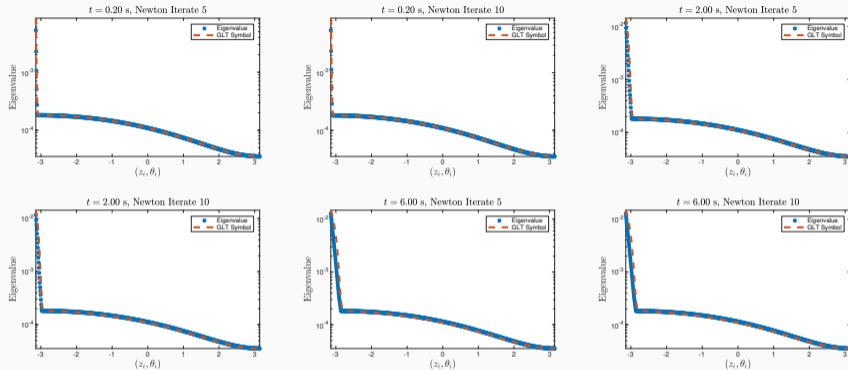
$$f(\mathbf{x}, \theta) = C \rho \phi s'(\mathbf{p}^{(k,j)}(\psi(\mathbf{x}))) + K(\mathbf{p}^{(k,j)}(\psi(\mathbf{x}))) (8 - 2 \cos(\theta_1) - 2 \cos(\theta_2) - 2 \cos(\theta_3)),$$

where $\mathbf{x} \in [0, 1]^3$, $\theta \in [-\pi, \pi]^3$, $\psi(\mathbf{x})$ is the function mapping $[0, 1]^3$ cube to the physical domain, and $C = \lim_{\mathbf{N}, N_T \rightarrow \infty} \frac{h}{\Delta t}$.

Take home messages:

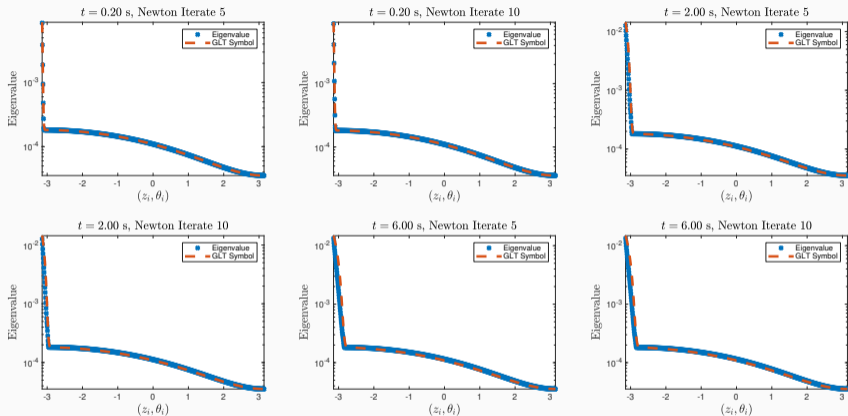
- ! Eigenvalue distribution is determined by the **diffusive part**,
- ! **III-conditioning** comes both from diffusive behavior and decay to zero of $K(p)$,
- ! We use the “diffusive part” of $\{J_{\mathbf{N}}^{(k,j)}\}_{\mathbf{N}}$ as $\{M_{\mathbf{N}}\}_{\mathbf{N}}$ (*throw away the transport term*).

Spectral Analysis: a visual representation



Arithmetic mean. Comparison of the eigenvalues and spectral symbol with $h_z = 40/(N - 1)$, $\Delta t = 0.1$, and $N = 800$ on different time steps and for different iterates of the Newton method \Rightarrow it works also far from the asymptotic regime.

Spectral Analysis: a visual representation




Upstream mean. Comparison of the eigenvalues and spectral symbol with $h_z = 40/(N - 1)$, $\Delta t = 0.1$, and $N = 800$ on different time steps and for different iterates of the Newton method \Rightarrow it works also far from the asymptotic regime.

A link with the literature

- ⚠ The idea of using the diffusive part to precondition is *somewhat natural*, see, e.g., [Jones & Woodward, 2001], **but** now we have a proof of why it works,
- ⚙ The schematic of the proof works for **different choices of the fluxes** at the interfaces,
- 🔧 We use the **Generalized Locally Toeplitz** machinery to achieve the formal result; see the books/papers by [Serra & Garoni 2017], [Barbarino, Serra, Garoni 2020].

But

- 🔧 We still need to find a way to apply $\{M_{\mathbf{N}}^{-1}\}_{\mathbf{N}}$ sequence: but now the sequence is **guaranteed** to be **SPD**!
- 📍 As per the discussed  plan, we will use an **Algebraic MultiGrid** method to generate a $\{\tilde{M}_{\mathbf{N}}^{-1}\}_{\mathbf{N}} \approx \{M_{\mathbf{N}}^{-1}\}_{\mathbf{N}}$ sequence.

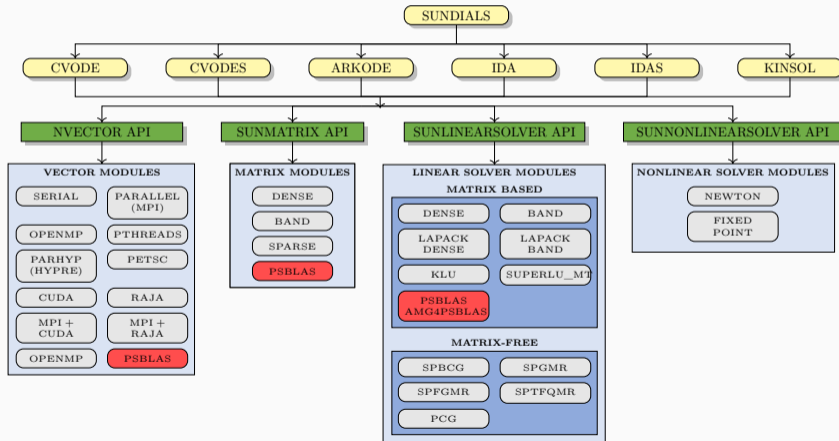
Two central libraries PSBLAS and **AMG4PSBLAS**:

- **Domain decomposition** preconditioners of Schwartz type
- Algebraic multigrid with **aggregation schemes**
 - Parallel coupled Weighted Matching Based Aggregation
 - Smoothed Aggregation (Vaněk, Mandel, Brezina)
- **Parallel Smoothers** (Block-Jacobi, DD-Schwartz, Hybrid-GS/SGS/FBGS, ℓ_1 variants) that can be coupled with specialized block (approximate) solvers MUMPS, SuperLU, incomplete factorizations ((H)AINV, (H)INVK/L, (H)ILU-type)
- V-Cycle, W-Cycle, K-Cycle



</> Opensource code, BSD3 License, free as in free  and as in free .

- ⚠ To implement the Newton part of the Newton-Krylov solver we implemented an extension to the SUNDIALS KINSOL package.



Preconditioners: generating the $\{\tilde{M}_N\}_N$ sequence

Among the different methods implemented in PSCToolkit we apply:

AS: One-level Additive Schwarz preconditioner.

DSVMB: the **smoothed aggregation scheme** introduced in (Vaněk, Mandel, Brezina 1996), and applied in a **parallel setting** by a **decoupled approach**, where each process applies the coarsening algorithm to its subset of dofs, ignoring interactions with dofs owned by other processes (D'Ambra, di Serafino, Filippone).

SMATCH: the **smoothed aggregation scheme** introduced in (D'Ambra, Vassilevski, 2013; D'Ambra, Filippone, Vassilevski 2018; D'Ambra, D. Filippone, 2021). It relies on a **parallel coupled aggregation** of dofs based on a **maximum weighted graph matching algorithm**, where the maximum size of aggregates can be chosen in a flexible way by a user-defined parameter.

Preconditioners: generating the $\{\tilde{M}_N\}_N$ sequence

Among the different methods implemented in PSCToolkit we apply:

AS: One-level Additive Schwarz preconditioner. AS uses one layer of mesh points in each direction as overlap among the subdomains, each of them assigned to different processes, and applies an Incomplete LU factorization with no fill-in for computing the local subdomain matrix inverses


AMG Setup Symmetric **V-cycle** with 1 iteration of **hybrid backward/forward Gauss-Seidel** as **pre/post-smoother** at the intermediate levels. As **coarsest-level solver** we use a parallel iterative procedure based on the **preconditioned Conjugate Gradient** method with **block-Jacobi** as preconditioner, where **ILU with 1 level of fill-in** is applied on the local diagonal blocks.

Test problem and solver details

- Richards equation discretized with upstream averages on a parallelepipedal domain Ω of size $[0, L_x] \times [0, L_y] \times [0, L]$,
- **Boundary conditions:** water at height $z = L$ such that the pressure head becomes zero in a square region at the center of the top layer, $(\frac{a}{4} \leq x \leq \frac{3a}{4}, \frac{b}{4} \leq y \leq \frac{3b}{4})$, and is fixed to the value $h = h_r$ on all the remaining boundaries,
- **Initial condition:** $p(x, y, z, 0) = h_r$. In all cases we run the simulation for $t \in [0, 2]$ and $N_t = 10$.

Solver options

PSBLAS-based Right preconditioned Restarted GMRES(10) with restarting step equal to 10. Relative residual stop $\|J\mathbf{d}_r + \Phi\| < \eta\|\Phi\|$ with $\eta = 10^{-7}$ or maximum number of iterations: 200.

 We update the AMG hierarchies by reusing the projectors and rebuilding the smoothers.

Test problem and solver details

- Richards equation discretized with upstream averages on a parallelepipedal domain Ω of size $[0, L_x] \times [0, L_y] \times [0, L]$,
- **Boundary conditions:** water at height $z = L$ such that the pressure head becomes zero in a square region at the center of the top layer, $(\frac{a}{4} \leq x \leq \frac{3a}{4}, \frac{b}{4} \leq y \leq \frac{3b}{4})$, and is fixed to the value $h = h_r$ on all the remaining boundaries,
- **Initial condition:** $p(x, y, z, 0) = h_r$. In all cases we run the simulation for $t \in [0, 2]$ and $N_t = 10$.

Strong scaling

Parallelepiped $[0, 64] \times [0, 64] \times [0, 1]$, $N_x = N_y = 800$ mesh points in the x and y directions, $N_z = 40$ mesh points in the vertical direction. **Total number of 20 millions dofs**, on a **number of computational cores** from 1 to 256.

Test problem and solver details

- Richards equation discretized with upstream averages on a parallelepipedal domain Ω of size $[0, L_x] \times [0, L_y] \times [0, L]$,
- **Boundary conditions:** water at height $z = L$ such that the pressure head becomes zero in a square region at the center of the top layer, $(\frac{a}{4} \leq x \leq \frac{3a}{4}, \frac{b}{4} \leq y \leq \frac{3b}{4})$, and is fixed to the value $h = h_r$ on all the remaining boundaries,
- **Initial condition:** $p(x, y, z, 0) = h_r$. In all cases we run the simulation for $t \in [0, 2]$ and $N_t = 10$.

Weak scaling

Parallelepiped $\Omega(np) = [0, 2^p \times 4.0] \times [0, 2^q \times 4.0] \times [0, 1.0]$ splitted on $np = p \times q$ processes for increasing $p = 0, \dots, 7$, $q = 0, \dots, 6$, corresponding mesh with $N(p \times q) = (2^p N_x, 2^q N_y, N_z)$ dofs, where $N_x = N_y = 50$, and $N_z = 40$. **Global size** up to about 829 **millions of dofs** on **8192 processes**.

Test problem and solver details

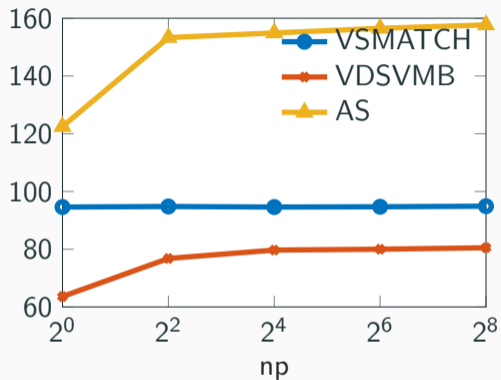
- Richards equation discretized with upstream averages on a parallelepipedal domain Ω of size $[0, L_x] \times [0, L_y] \times [0, L]$,
- **Boundary conditions:** water at height $z = L$ such that the pressure head becomes zero in a square region at the center of the top layer, $(\frac{a}{4} \leq x \leq \frac{3a}{4}, \frac{b}{4} \leq y \leq \frac{3b}{4})$, and is fixed to the value $h = h_r$ on all the remaining boundaries,
- **Initial condition:** $p(x, y, z, 0) = h_r$. In all cases we run the simulation for $t \in [0, 2]$ and $N_t = 10$.

Machine & Environment

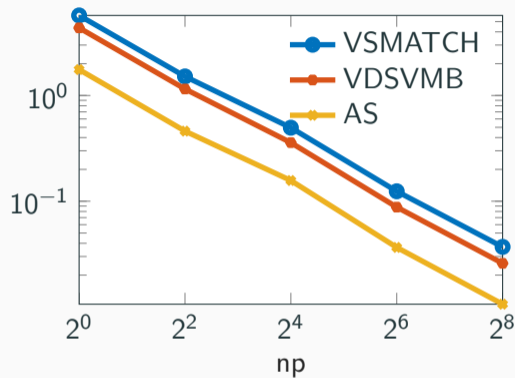
All the experiments are executed on the CPU cores, with no usage of hyperthreading, of the *Marconi-100 supercomputer* (18th in the November 2021 TOP500 list).

Compilers: gnu/8.4.0; **Libraries:** openmpi/4.0.3, openblas/0.3.9, PSBLAS 3.7.0.2 and AMG4PSBLAS 1.0.

🔧 Strong scaling: 20 Million Dofs on 1 to 256 cores.



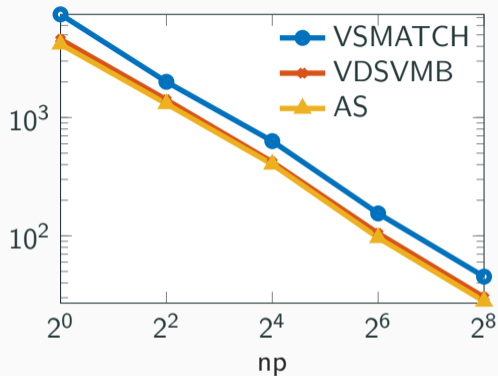
Avg. number of linear iterations



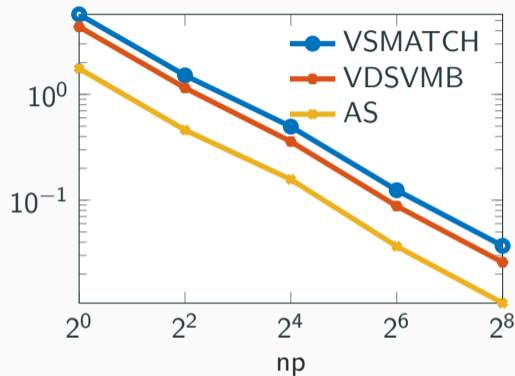
Time per iteration (s)

Efficiency: ranging from 59% to 66%.

🔧 Strong scaling: 20 Million Dofs on 1 to 256 cores.



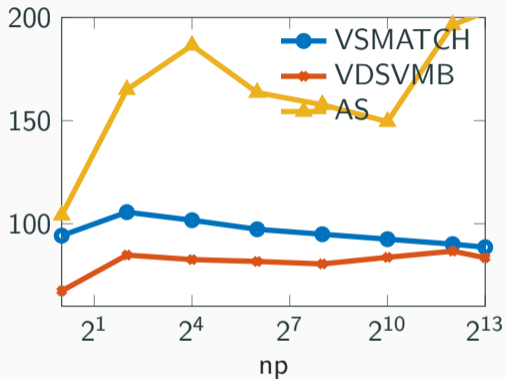
Total time to solution (s)



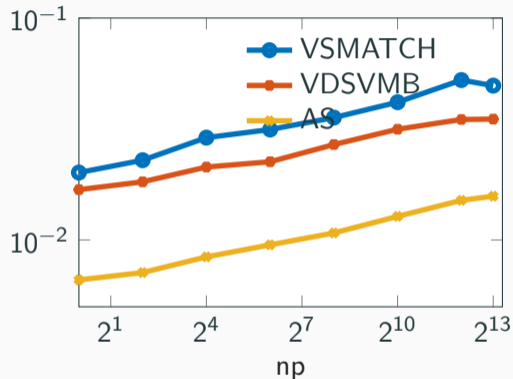
Time per iteration (s)

Efficiency: ranging from 59% to 66%.

Weak scaling: 829 Milion Dofs on 1 to 8192 cores.

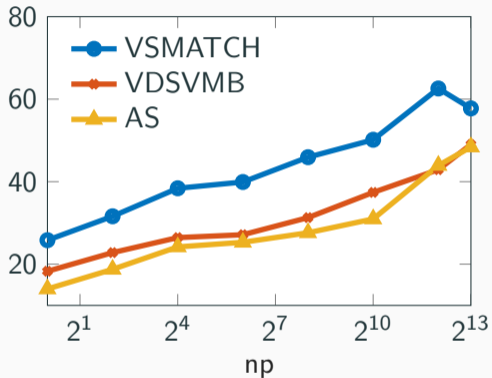


Avg. number of linear iterations

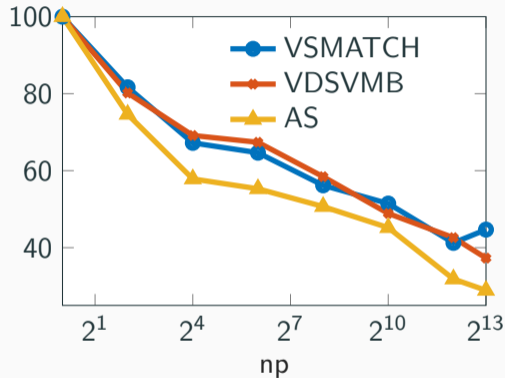


Time per iteration (s)

Weak scaling: 829 Milion Dofs on 1 to 8192 cores.



Total time to solution (s)




Efficiency (%)




Concluding remarks

- ✓ Determined the **spectral properties** of the Jacobian matrix sequence for a **range** of **discretization choices**,
- ✓ We now have a **formal proof** of the intuition behind the “*preconditioning with the diffusive part*” idea,

Concluding remarks

- ✓ Determined the **spectral properties** of the Jacobian matrix sequence for a **range** of **discretization choices**,
- ✓ We now have a **formal proof** of the intuition behind the “*preconditioning with the diffusive part*” idea,
-  **Performances** of our strategy are quite **promising** in view of exploring **extreme scalability**.

Concluding remarks

- ✓ Determined the **spectral properties** of the Jacobian matrix sequence for a **range** of **discretization choices**,
- ✓ We now have a **formal proof** of the intuition behind the “*preconditioning with the diffusive part*” idea,
-  **Performances** of our strategy are quite **promising** in view of exploring **extreme scalability**.
-  PSCToolkit has also a GPU support and we would like to explore also in that direction.
-  We have an integration in PARFLOW planned to treat more realistic and *coupled* problems.

Concluding remarks

- ✓ Determined the **spectral properties** of the Jacobian matrix sequence for a **range** of **discretization choices**,
- ✓ We now have a **formal proof** of the intuition behind the “*preconditioning with the diffusive part*” idea,
- 👤 **Performances** of our strategy are quite **promising** in view of exploring **extreme scalability**.
- 📱 PSCToolkit has also a GPU support and we would like to explore also in that direction.
- 🔗 We have an integration in PARFLOW planned to treat more realistic and *coupled* problems.
 - 📄 Bertaccini, D., D'Ambra, P., D., F., & Filippone, S. (2021). *Why diffusion-based preconditioning of Richards equation works: spectral analysis and computational experiments at very large scale*. *arXiv preprint arXiv: 2112. 05051*

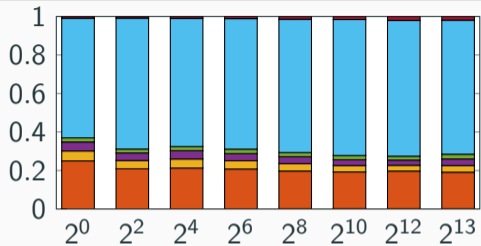
Thank you!

Strong scaling: 20 Milion Dofs on 1 to 256 cores.

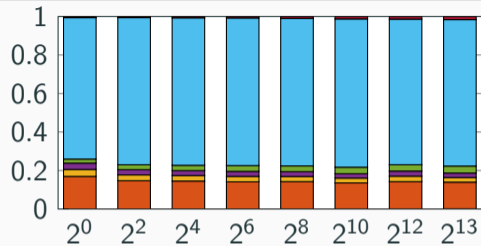
	VDSVBM		VSMATCH		AS	
np	N Jac.s	NLin It.s	N Jac.s	NLin It.s	N Jac.s	NLin It.s
1	3	36	3	38	3	43
4	3	37	3	38	4	39
16	3	37	3	38	4	39
64	3	37	3	38	4	39
256	3	37	3	38	4	39

Table 1: Strong scaling. Number of nonlinear iterations (NLin It.s), and number of computed Jacobians (N Jac.s) for the three preconditioners.

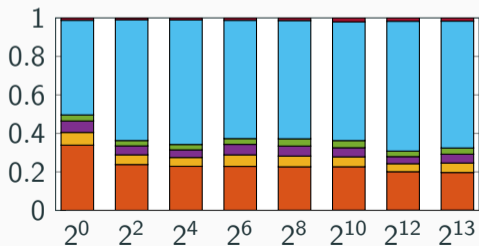
🔧 Weak scaling: 829 Milion Dofs on 1 to 8192 cores.



VDSVMB



VSMATCH



AS

