Scalability Results for the Solution of the Richards Equation as in ParFlow

Fabio Durastante

EoCoE Consortium Meeting

Decembre 13-15, 2021

Dipartimento di Matematica, Università di Pisa
`fabio.durastante@unipi.it`

Consiglio Nazionale delle Ricerche
Istituto per le Applicazioni del Calcolo "M. Picone" – (IAC-CNR)

Collaborators, funding and acknowledgments

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

Stefan Kollet,
Jülich Forschungszentrum,
Institute of Bio- and Geosciences.

Thanks for the helpful discussions on the model and discretization in PARFLOW!

EoCoE
Project ID: 824158
European Commission
Horizon 2020
European Union funding
for Research & Innovation

The Richards Equation as considered in ParFlow

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**

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

- $s(p)$ 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

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

  - $K(p)$ 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 complete 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]

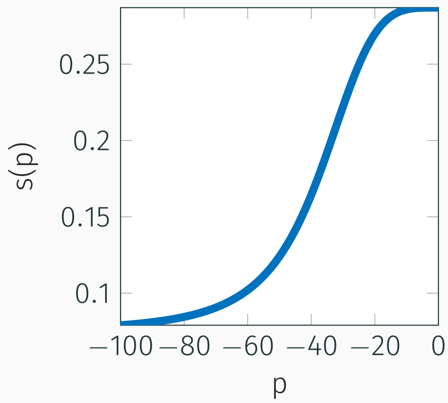
$$s(p) = \frac{\alpha(s_s - s_r)}{\alpha + |p|^\beta} + s_r, \text{ and } K(p) = K_s \frac{a}{a + |p|^\gamma},$$

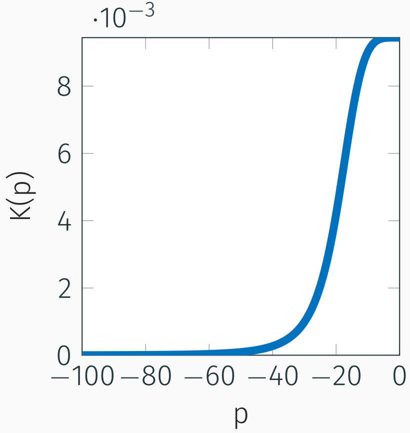
where

- 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 complete 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 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**:

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

Cell-centered finite difference discretization

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

with

$$\begin{aligned}
q_{i+1/2,j,k}^{(l)} &= -{}^{\mathrm{AV}}K_{i+1,i}^{(l)}\left(\frac{p_{i+1,j,k}^{(l)} - p_{i,j,k}^{(l)}}{h_x^2}\right), q_{i-1/2,j,k}^{(l)} = -{}^{\mathrm{AV}}K_{i-1,i}^{(l)}\left(\frac{p_{i,j,k}^{(l)} - p_{i-1,j,k}^{(l)}}{h_x^2}\right), \\
q_{i,j+1/2,k}^{(l)} &= -{}^{\mathrm{AV}}K_{j+1,j}^{(l)}\left(\frac{p_{i,j+1,k}^{(l)} - p_{i,j,k}^{(l)}}{h_y^2}\right), q_{i,j-1/2,k}^{(l)} = -{}^{\mathrm{AV}}K_{j-1,j}^{(l)}\left(\frac{p_{i,j,k}^{(l)} - p_{i,j-1,k}^{(l)}}{h_y^2}\right), \\
q_{i,j,k+1/2}^{(l)} &= -{}^{\mathrm{AV}}K_{k+1,k}^{(l)}\left(\frac{p_{i,j,k+1}^{(l)} - p_{i,j,k}^{(l)}}{h_z^2}\right) - \frac{K(p_{i,j,k+1})}{2h_z}, \\
q_{i,j,k-1/2}^{(l)} &= -{}^{\mathrm{AV}}K_{k-1,k}^{(l)}\left(\frac{p_{i,j,k}^{(l)} - p_{i,j,k-1}^{(l)}}{h_z^2}\right) - \frac{K(p_{i,j,k-1})}{2h_z},
\end{aligned}$$

The Newton method and the sequence of the Jacobians

- Newton step for the solution, at each time step, of the nonlinear systems,
- The Jacobian matrix $J = J_{\mathbf{\Phi}}$ can then be computed in closed form,
- At the core of the (distributed) parallel solution resides the solution of the (right) preconditioned linear system

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

What did we do in `https://arxiv.org/abs/2112.05051`:

- Describe the asymptotic spectral properties of the sequence $\{J_{\mathrm{N}}\}_{\mathrm{N}}$,
- Analyze the impact of (some) of the different **choices for the interface mean**,
- Use this information to get a matrix sequence $\{M_{\mathrm{N}}\}_{\mathrm{N}}$ for preconditioning $\{J_{\mathrm{N}}\}_{\mathrm{N}}$,
- Approximate such matrix sequence by a (parallel) AMG method to efficiently solve the systems.

The Newton method and the sequence of the Jacobians

- Newton step for the solution, at each time step, of the nonlinear systems,
- The Jacobian matrix $J = J_{\mathbf{\Phi}}$ can then be computed in closed form,
- At the core of the (distributed) parallel solution resides the solution of the (right) preconditioned linear system

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

What did we do in `https://arxiv.org/abs/2112.05051`:

- Use this information to get a matrix sequence $\{M_{\mathtt{N}}\}_{\mathtt{N}}$ for preconditioning $\{J_{\mathtt{N}}\}_{\mathtt{N}}$,
- Approximate such matrix sequence by a (parallel) AMG method to efficiently solve the systems.

I'll **focus** here on the **implementation aspects**, for the spectral analysis and the other mathematical information: `https://arxiv.org/abs/2112.05051`

The Theoretical sequence of preconditioners (cont'd)

- The theoretical analysis tells us that we can use the discretization of the diffusion operator to precondition. This is *somewhat natural*, see, e.g., [Jones & Woodward, 2001], **but** now we **have a proof** of why it works,
- The organization of the proof works for **different choices of the fluxes** at the interfaces,
- We use the **G**eneralized **L**ocally **T**oeplitz 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. Even if the sequence is simpler.
- Use an Algebraic Multigrid Algorithm to generate a $\{\tilde{M}_{\mathbf{N}}^{-1}\}_{\mathbf{N}}$ sequence.

What do we ask to it?

Solve the preconditioned system:

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

with matrix $\tilde{M}^{-1} \approx J^{-1}$ (right preconditioner) such that:

Algorithmic scalability $\max_i \lambda_i(\tilde{M}^{-1}J) \approx 1$ being independent of $\mathbf{N}$,

Linear complexity the action of $\tilde{M}^{-1}$ costs as little as possible, the best being $\mathcal{O}(\mathbf{N})$ flops,

Implementation scalability in a massively parallel computer, $\tilde{M}^{-1}$ should be composed of local actions, performance should depend linearly on the number of processors employed.

Observe that by the GLT analysis, we know that $\max_i \lambda_i(M^{-1}J) \approx 1$, thus if our multigrid hierarchy is “good enough” we can achieve a “near enough” result with it.

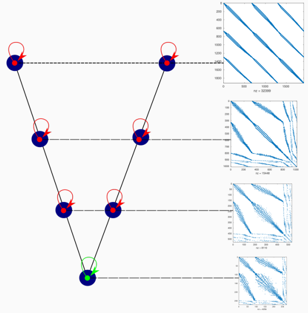
An Algebraic Multigrid Approximation of $\{M_{\mathrm{N}}^{-1}\}_{\mathrm{N}}$

Given Matrix $M_{\mathrm{N}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ SPD

Wanted Iterative method $\tilde{M}$ to precondition a Krylov iterative method:

- Hierarchy of systems
  $$R_l \mathbf{x}_= \mathbf{b}_l, l = 0, \dots, \mathrm{nlev}$$
- Transfer operators:
  $$P_{l+1}^{l} : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_l}$$

Missing Structural/geometric infos



Smoother

$$R_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$$

"High frequencies"

Prolongator

$$P_{l+1}^{l} : \mathbb{R}^{n_l} \to \mathbb{R}^{n_{l+1}}$$

"Low frequencies"

Complementarity of Smoother and Prolongator

Parallel Sparse Computation Toolkit – `psctoolkit.github.io`

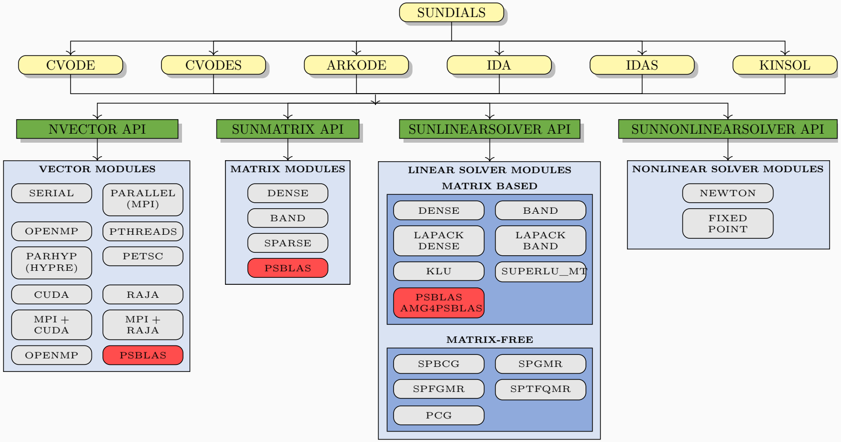
Two central libraries PSBLAS and AMG4PSBLAS:

- Algebraic multigrid with aggregation schemes
  - Parallel coupled weighted matching based aggregation
  - Parallel decoupled 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 (AINV, INVK/L, ILU-type)
- V-Cycle, W-Cycle, K-Cycle

D'Ambra, P., F. D., and S. Filippone. "AMG preconditioners for linear solvers towards extreme scale." SIAM J. Sci. Comp. 43.5 (2021): S679-S703.

The KINSOL Software Framework

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



From KINSOL to ParFlow

- Wrapping of PSCToolkit *distributed sparse linear algebra* in KINSOL
  - NVECTOR: distributed vectors with all relevant operations (`axpy`, `norms`, `dot`, integrated actions for group of vectors, ...)
  - SUNMatrix: distributed matrix for all the formats in PSBLAS (`CSR`, `CSC`, `COO`, `HYB`, ...) and all the relevant operators (`spmv`, `matrix shift`, ...)
  - SUNLinSol: interface to *all* the Krylov **linear solvers** in PSBLAS (`CG`, `GMRES`, `BiCGStab`, ...) and all the **preconditioner** that can be used (or added in future) to AMG4PSBLAS (Algebraic Multigrid with different aggregation strategies, Domain Decomposition techniques)

From KINSOL to ParFlow

- Wrapping of PSCToolkit *distributed sparse linear algebra* in KINSOL
  - NVECTOR: distributed vectors with all relevant operations (`axpy`, `norms`, `dot`, integrated actions for group of vectors, ...)
  - SUNMatrix: distributed matrix for all the formats in PSBLAS (CSR, CSC, COO, HYB, ...) and all the relevant operators (`spmv`, `matrix shift`, ...)
  - SUNLinSol: interface to *all* the Krylov **linear solvers** in PSBLAS (CG, GMRES, `BiCGStab`, ...) and all the **preconditioner** that can be used (or added in future) to AMG4PSBLAS (Algebraic Multigrid with different aggregation strategies, Domain Decomposition techniques)
- (PSCToolkit) ⇒ KINSOL ⇒ ParFlow

From KINSOL to ParFlow

- Wrapping of PSCToolkit *distributed sparse linear algebra* in KINSOL
  - NVECTOR: distributed vectors with all relevant operations (`axpy`, `norms`, `dot`, integrated actions for group of vectors, ...)
  - SUNMatrix: distributed matrix for all the formats in PSBLAS (`CSR`, `CSC`, `COO`, `HYB`, ...) and all the relevant operators (`spmv`, `matrix shift`, ...)
  - SUNLinSol: interface to *all* the Krylov **linear solvers** in PSBLAS (`CG`, `GMRES`, `BiCGStab`, ...) and all the **preconditioner** that can be used (or added in future) to AMG4PSBLAS (Algebraic Multigrid with different aggregation strategies, Domain Decomposition techniques)
- (PSCToolkit) ⇒ KINSOL ⇒ ParFlow
- KINSOL is used in many codes as the supplier of both linear and nonlinear solvers, this first integration is portable for other problems.

Problem and Machine

- Parallelepipedal domain Ω of size $[0, L_x] \times [0, L_y] \times [0, L]$,
- Water at height $z = L$ such that the pressure head becomes zero in a square region at the center of the top layer

$$p(x, y, L, t) = \frac{1}{\alpha} \ln \Big[\exp(\alpha h_r) + (1 - \exp(\alpha h_r)) \chi_{[\frac{a}{4}, \frac{3a}{4}]\times[\frac{b}{4}, \frac{3b}{4}]}(x, y, z)\Big],$$

- Initial condition is given by $p(x, y, z, 0) = h_r$,
- In all cases we run the simulation for $t \in [0, 2]$ and $N_t = 10$.

Marconi 100

(18th in 11/2021 TOP500)

IBM Power System AC922 nodes

2×16 *IBM POWER93* 3.1 GHz,

256 GB of RAM.

Dual-rail *Mellanox EDR*

Infiniband network by *IBM*

220/300 GB/s.

Preconditioners

	Multigrid		One-Level	
Cycle	1 sweep of V-cyle		Additive Schwarz	Type
Aggregation	Parallel **Decoupled** smoothed aggregation [Vaněk. Mandel, Brezina, 1996]	Parallel **Coupled** smoothed aggregation based on graph matching aggregate size: 8 [D'Ambra, Filippone, Vassilevski, 2018]	1 layer of mesh points in each grid direction	Overlap
Pre/post-smoother	1 iteration of hybrid backward/forward Gauss-Seidel		ILU(0)	Local solver
Coarsest solver	preconditioned **CG** method with ILU(1)-block-Jacobi preconditioner			
Label	VDSVMB	VSMATCH	AS	Label

Preconditioners

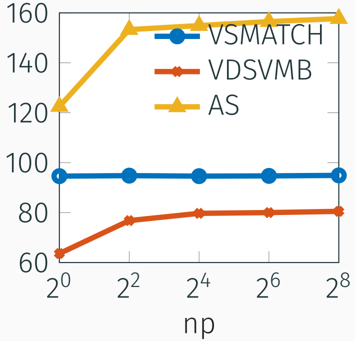
	Multigrid		One-Level	
Cycle	1 sweep of V-cyle		Additive Schwarz	Type
Aggregation	Parallel **Decoupled** smoothed aggregation [Vaněk. Mandel, Brezina, 1996]	Parallel **Coupled** smoothed aggregation based on graph matching aggregate size: 8 [D'Ambra, Filippone, Vassilevski, 2018]	1 layer of mesh points in each grid direction	Overlap
Pre/post-smoother	1 iteration of hybrid backward/forward Gauss-Seidel		ILU(0)	Local solver
Coarsest solver	preconditioned **CG** method with ILU(1)-block-Jacobi preconditioner			
Label	VDSVMB	VSMATCH	AS	Label

Preconditioners

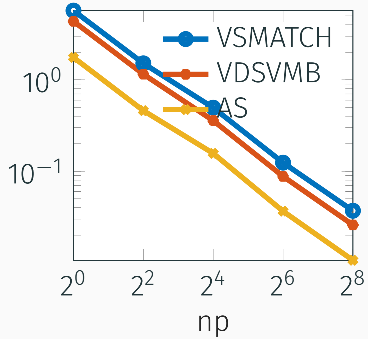
	Multigrid		One-Level	
Cycle	1 sweep of V-cyle		Additive Schwarz	Type
Aggregation	Parallel **Decoupled** smoothed aggregation [Vaněk. Mandel, Brezina, 1996]	Parallel **Coupled** smoothed aggregation based on graph matching aggregate size: 8 [D'Ambra, Filippone, Vassilevski, 2018]	1 layer of mesh points in each grid direction	Overlap
Pre/post-smoother	1 iteration of hybrid backward/forward Gauss-Seidel		ILU(0)	Local solver
Coarsest solver	preconditioned **CG** method with ILU(1)-block-Jacobi preconditioner			
Label	VDSVMB	VSMATCH	AS	Label

Strong Scalability Analysis

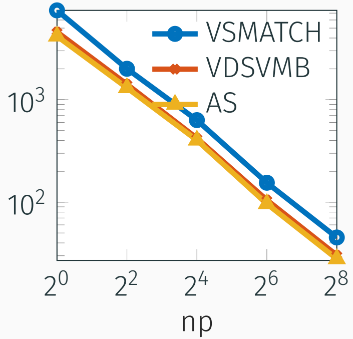
- Parallelepiped $[0,64] \times [0,64] \times [0,1]$, discretized with $N_x = N_y = 800$, and $N_z = 40 \Rightarrow$ 20 millions of dofs,
- Computational cores from 1 to 256, i.e., $np = 4^p$, $p = 0, \ldots, 4$,



Average number of linear iterations



Average time per linear iteration T (s)



Total solution time T (s)

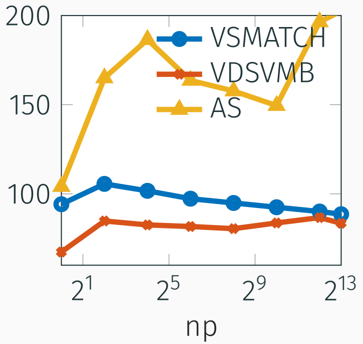
Strong Scalability Analysis

	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

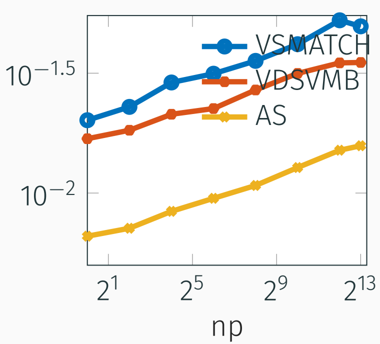
Number of **nonlinear iterations** (NLin It.s), and number of **computed Jacobians** (N Jac.s) for the three preconditioners.

Weak scalability analysis

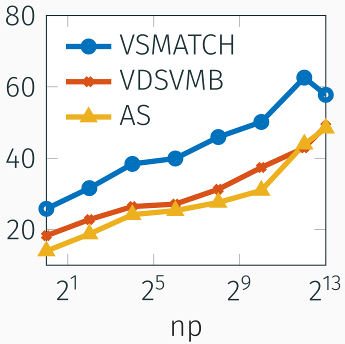
- $N_x = N_y = 50$, and $N_z = 40$, $\Omega(np) = [0, 2^p \times 4.0] \times [0, 2^q \times 4.0] \times [0, 1.0]$
- $np = p \times q$ processes, $p = 0, \dots, 7$, $q = 0, \dots, 6$, and a corresponding mesh $N(p \times q) = (2^p N_x, 2^q N_y, N_z) \Rightarrow$ 820 millions of dofs.



Average number of linear iterations



Average time per linear iteration T (s)



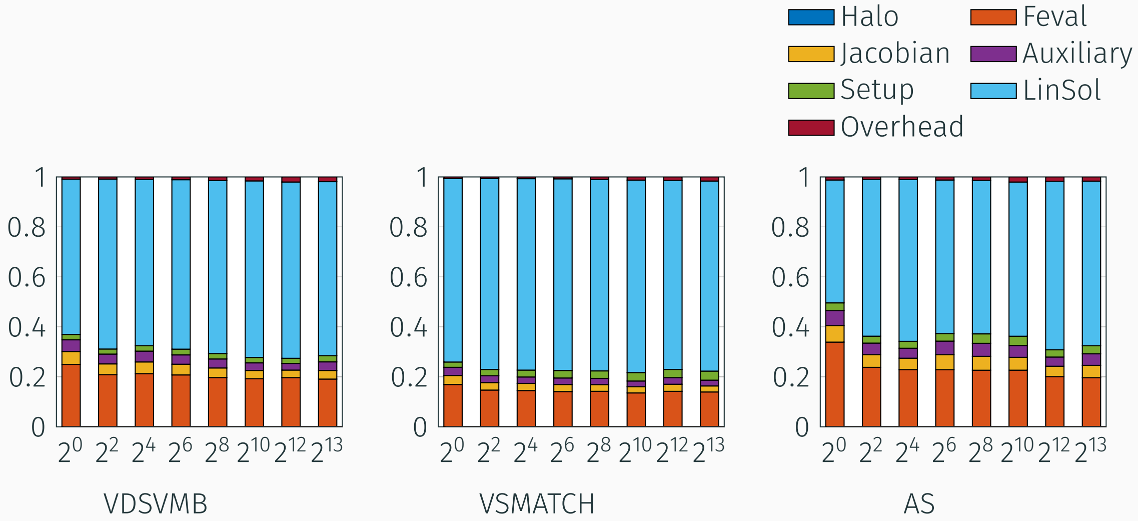
Total solution time T (s)

Weak scalability analysis

	VDSVBM		VSMATCH		AS	
np	N Jac.s	NLin It.s	N Jac.s	NLin It.s	N Jac.s	NLin It.s
1	3	37	3	36	3	40
4	3	38	3	38	3	36
16	3	38	3	38	3	40
64	3	37	3	38	4	37
256	3	37	3	38	4	39
1024	3	39	3	38	4	41
4096	3	41	3	38	4	47
8192	3	40	3	38	4	48

Number of **nonlinear iterations** (NLin It.s), and number of **computed Jacobians** (N Jac.s) for the three preconditioners.

Weak scalability analysis - Time Fractions



Conclusions and future perspectives

We focused on **two main objectives**

- ✓ prove some asymptotic spectral properties of the sequence of Jacobian matrices generated discretizing the Richards equation;
- ✓ prove the efficiency, flexibility and robustness of a software framework for parallel sparse matrix computations.

Our plans for the future

- 🔧 extension of the PSCToolkit interface to KINSOL, in order to use the ability of the PSCToolkit linear solvers in exploiting GPU architectures;
- 🔧 integration of the software stack into the PARFLOW code for realistic simulations in hydrological applications.

Thank you!

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 $^{\mathrm{ARIT}}K = (K_U + K_L)/2,$

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

  upstream-weighted mean

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

  integral mean

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

- A combination of the above in the different directions

An Algebraic Multigrid based on Matching

Given $\mathbf{w} \in \mathbb{R}^n$, let $P \in \mathbb{R}^{n \times n_c}$ and $P_f \in \mathbb{R}^{n \times n_f}$ be a prolongator and a complementary prolongator, such that:

$$\mathbb{R}^n = \text{Range}(P) \oplus^{\perp} \text{Range}(P_f), \quad n = n_c + n_f$$

$\mathbf{w} \in \text{Range}(P)$: coarse space

$\text{Range}(P_f)$: complementary space

$$[P, P_f]^T M [P, P_f] = \begin{pmatrix} P^T M P & P^T M P_f \\ P_f^T M P & P_f^T M P_f \end{pmatrix} = \begin{pmatrix} M_c & M_{cf} \\ M_{fc} & M_f \end{pmatrix}$$

M_c: coarse matrix

M_f: hierarchical complement

Sufficient condition for efficient coarsening

$M_f = P_f^T M P_f$ as well conditioned as possible, i.e.,

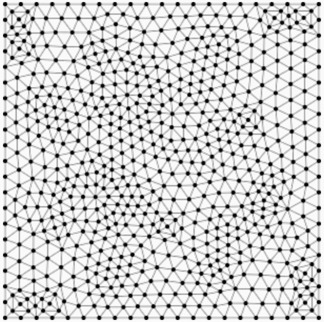
Convergence rate of *compatible relaxation*: $\rho_f = \|I - R_f^{-1} M_f\|_{M_f} \ll 1$

But *how* we achieve it?

Weighted graph matching
Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix M), and a weight vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{M}$:

$$(\hat{M})_{i,j} = \hat{m}_{i,j} = 1 - \frac{2m_{i,j}w_iw_j}{m_{i,i}w_i^2 + m_{j,j}w_j^2},$$

- a *matching* $\mathcal{M}$ is a set of pairwise non-adjacent edges, containing no loops;
- a maximum product matching if it maximizes the product of the weights of the edges $e_{i\mapsto j}$ in it.



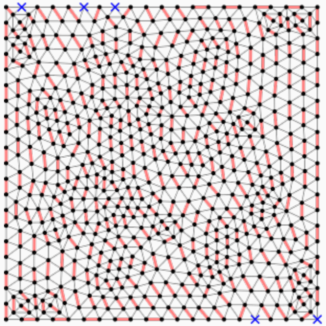
But *how* we achieve it?

Weighted graph matching

Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix M), and a weight vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{M}$:

$$(\hat{M})_{i,j} = \hat{m}_{i,j} = 1 - \frac{2m_{i,j}w_iw_j}{m_{i,i}w_i^2 + m_{j,j}w_j^2},$$

- a *matching* $\mathcal{M}$ is a set of pairwise non-adjacent edges, containing no loops;
- a maximum product matching if it maximizes the product of the weights of the edges $e_{i\mapsto j}$ in it.



We divide the index set into matched vertexes $\mathcal{I} = \bigcup_{i=1}^{n_p} \mathcal{G}_i$, with $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ if $i \neq j$, and unmatched vertexes, i.e., n_s singletons G_i.

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \underbrace{\left[\begin{array}{cc} \underbrace{\begin{bmatrix} \mathbf{w}_{e_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{w}_{e_{n_p}} \end{bmatrix}}_{n_p} 2n_p & \mathbf{0} \\ \mathbf{0} & \underbrace{\begin{bmatrix} w_1/|w_1| & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_{n_s}/|w_{n_s}| \end{bmatrix}}_{n_s} n_s \end{array}\right]}_{n_c = n_p + n_s = J} n = 2n_p + n_s$$

$$= \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \ldots, \mathbf{p}_J], \qquad \mathbf{w}_e = \frac{1}{\sqrt{w_i^2 + w_j^2}} \begin{bmatrix} w_i \\ w_j \end{bmatrix}.$$

$\Rightarrow$ The $\mathcal{M}$ on $\hat{M}$ produces M_f with diagonal entries $\hat{a}_{ij}$ for $(i,j) \in \mathcal{M}$ of maximal product.

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \ldots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - \tilde{M}_l M_l = (I - (R_l)^{-T} M_l)(I - P_l \tilde{M}_{l+1} (P_l)^T M_l)(I - R_l^{-1} M_l) \quad \forall l < nl,$$

where $M_{l+1} = (P_l)^T M_l P_l$ for $l = 0, \ldots, nl - 1$.

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \dots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - \tilde{M}_l M_l = (I - (R_l)^{-T} M_l)(I - P_l \tilde{M}_{l+1} (P_l)^T M_l)(I - R_l^{-1} M_l) \quad \forall l < nl,$$

where $M_{l+1} = (P_l)^T M_l P_l$ for $l = 0, \dots, nl - 1$.

- To increase dimension reduction we can perform more than one sweep of matching per step,

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \dots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - \tilde{M}_l M_l = (I - (R_l)^{-T} M_l)(I - P_l \tilde{M}_{l+1} (P_l)^T M_l)(I - R_l^{-1} M_l) \quad \forall l < nl,$$

where $M_{l+1} = (P_l)^T M_l P_l$ for $l = 0, \dots, nl - 1$.

- To increase dimension reduction we can perform more than one sweep of matching per step,
- To increase regularity of P_l we can consider a smoothed prolongator by applying a Jacobi smoother,

$$P_l^s = (I - \omega D_l^{-1} M_l) P_l, \text{ for } D_l = \mathrm{diag}(M_l).$$

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \ldots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - \tilde{M}_l M_l = (I - (R_l)^{-T} M_l)(I - P_l \tilde{M}_{l+1} (P_l)^T M_l)(I - R_l^{-1} M_l) \;\; \forall l < nl,$$

where $M_{l+1} = (P_l)^T M_l P_l$ for $l = 0, \ldots, nl - 1$.

- To increase dimension reduction we can perform more than one sweep of matching per step,
- To increase regularity of P_l we can consider a smoothed prolongator by applying a Jacobi smoother,
- To increase the robustness we can use a non stationary solver as smoother.

The Theoretical sequence of preconditioners

To devise the preconditioners for these problems we want to **leverage on spectral information** about the sequence $\{J_{\mathbf{N}}\}_{\mathbf{N}}$

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

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

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

The Theoretical sequence of preconditioners

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 (GLT symbol)

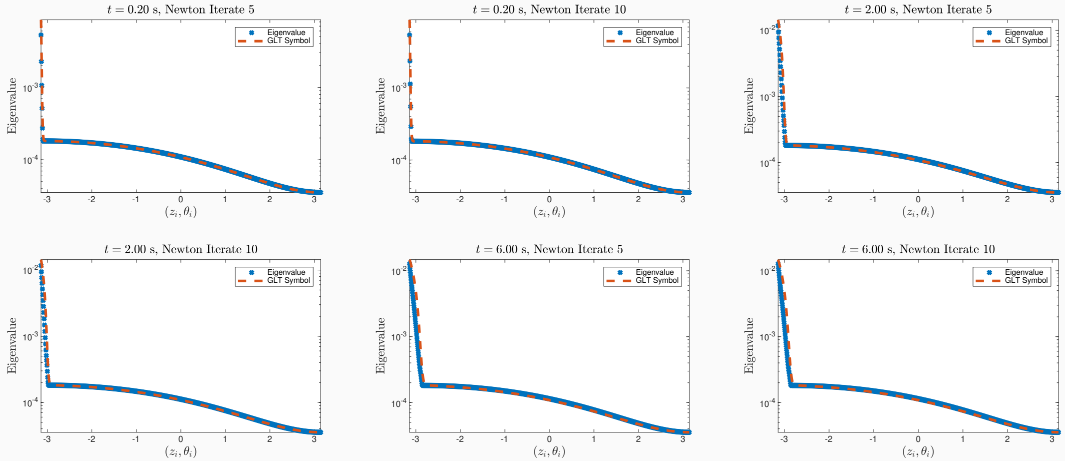
$$f(\mathbf{x},\theta) = C\rho\phi s'(\mathbf{p}^{(k,j)}(\psi(\mathbf{x}))) + K(\mathbf{p}^{(k,j)}(\psi(\mathbf{x})))(8 - 2\cos(\theta_1) - 2\cos(\theta_2) - 2\cos(\theta_3)),$$

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 $C = \lim_{\mathbf{N},N_T\to\infty} \frac{\mathrm{h}}{\Delta t}$.

Take-home messages:

- Eigenvalue distribution is determined by the diffusive part,
- **Ill-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 preconditioner (throw away the transport term).

The Theoretical sequence of preconditioners (an example)



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.