Received: 15 July 2022 | Revised: 8 May 2023 | Accepted: 27 June 2023

DOI: 10.1002/nla.2523

RESEARCH ARTICLE

WILEY

Why diffusion-based preconditioning of Richards equation works: Spectral analysis and computational experiments at very large scale

Daniele Bertaccini[1,2] | **Pasqua D'Ambra**[2] | **Fabio Durastante**[2,3] | **Salvatore Filippone**[2,4]

[1]Dipartimento di Matematica, Università di Roma "Tor Vergata", Rome, Italy

[2]Istituto per le Applicazioni del Calcolo "Mauro Picone", Consiglio Nazionale delle Ricerche, Naples, Italy

[3]Dipartimento di Matematica, Università di Pisa, Pisa, Italy

[4]Department of Civil and Computer Engineering, Università di Roma "Tor Vergata", Rome, Italy

Correspondence
Fabio Durastante, Dipartimento di Matematica, Università di Pisa, Largo Bruno Pontecorvo 5, Pisa (PI) 56127, Italy.
Email: fabio.durastante@unipi.it

Funding information
H2020-INFRAEDI-2018-2020, Grant/Award Number: 824158

Abstract
We consider here a cell-centered finite difference approximation of the Richards equation in three dimensions, averaging for interface values the hydraulic conductivity $K = K(p)$, a highly nonlinear function, by arithmetic, upstream and harmonic means. The nonlinearities in the equation can lead to changes in soil conductivity over several orders of magnitude and discretizations with respect to space variables often produce stiff systems of differential equations. A fully implicit time discretization is provided by *backward Euler* one-step formula; the resulting nonlinear algebraic system is solved by an inexact Newton Armijo–Goldstein algorithm, requiring the solution of a sequence of linear systems involving Jacobian matrices. We prove some new results concerning the distribution of the Jacobians eigenvalues and the explicit expression of their entries. Moreover, we explore some connections between the saturation of the soil and the ill conditioning of the Jacobians. The information on eigenvalues justifies the effectiveness of some preconditioner approaches which are widely used in the solution of Richards equation. We also propose a new software framework to experiment with scalable and robust preconditioners suitable for efficient parallel simulations at very large scales. Performance results on a literature test case show that our framework is very promising in the advance toward realistic simulations at extreme scale.

KEYWORDS
algebraic multigrid, high performance computing, Richards equation, spectral analysis

1 | INTRODUCTION

Groundwater flow in the unsaturated zone is a highly nonlinear phenomenon that can be modeled by the Richards equation, and there is a significant amount of research concerning different formulations and algorithms for calculating the flow of water through unsaturated porous media.[1-5]

The Richards equation is a time-dependent Partial Differential Equation (PDE) whose discretization leads to large nonlinear systems of algebraic equations, which often include coefficients showing large variations over different orders of magnitude. Typically, the variation in the coefficients is due to the use of a geostatistical model for hydraulic conductivity, allowing changes of many orders of magnitude from one cell to the next (heterogeneity) as well as correlations of values in each direction (statistical anisotropy). High heterogeneity and anisotropy in the problem and the presence of strong nonlinearities in the equation's coefficients make the problem difficult to be solved numerically.

The main contribution of the present work is twofold. First, we investigate the spectral properties of the sequence of Jacobian matrices arising in the solution of the nonlinear systems of algebraic equations by a quasi-Newton method; these properties allow us to formulate a new theoretical justification for some preconditioning choices in solving Richards equation which are indeed current usual practices in the literature; to the best of our knowledge, this is the first theoretical formulation to support such common practices. Then, on the basis of the theoretical indications, we discuss a scalable and efficient parallel solution of a modified inexact quasi-Newton method with Krylov solvers and Armijo–Goldstein's line search, that is, Newton-like algorithms where the Newton correction linear equations are solved by a Krylov subspace method. To this aim, we interface the KINSOL package available from the Sundials project[6] with the most recent version of libraries for solving sparse linear systems by parallel Krylov solvers coupled with purely algebraic preconditioners,[7] currently being extended in some EU-funded projects.

The work is organized as follows. First, in Section 2 we discuss the discretization of the Richards equation by means of cell-centered finite differences. Then, in Section 3, we summarize the inexact quasi-Newton method as implemented in the KINSOL framework. In Section 4 we propose an analysis of some spectral properties of the sequence of Jacobian matrices produced by the underlying Newton method. In Section 5, we describe some of the main features of the PSCToolkit parallel software framework, which we use to implement the computational procedure described in this paper.

The information produced by the spectral analysis of the Jacobian matrices is used in Section 5.2 to devise a preconditioning strategy for the Krylov subspace method and to present the preconditioners used in this study. Some numerical examples highlighting the computational performance of the proposed methods are then presented in Section 6. Finally, in Section 7 we draw our conclusion and briefly discuss future extensions.

2 | FORMULATING THE DISCRETE PROBLEM

Flow in the vadose zone, which is also known as the *unsaturated* zone, has rather delicate aspects such as the parameters that control the flow, which depend on the saturation of the media, leading to the nonlinear problem described by the Richards equation. The flow can be expressed as a combination of Darcy's law and the principle of mass conservation by

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

in which $s(p)$ is the saturation at pressure head p of a fluid with density ρ in terrain with porosity ϕ, and q is the volumetric water flux. By using Darcy's law as

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

for $K(p)$ the hydraulic conductivity, and c the cosine of the angle between the downward z axis $\hat{z}$ and the direction of the gravity force, we obtain that the overall equation has a diffusion as well as an advection term, the latter being related to gravity. For the sake of simplicity, we consider cases in which $c = 1$, that is, the advection acts only in the z direction. There are two different forms of the Richards equation that differ in how they deal with the nonlinearity in the time derivative. The most popular form, which is the one considered here for a fluid that will always be water, permits to express the general equation as

$$\rho\, \phi \frac{\partial s(p)}{\partial t} - \nabla \cdot K(p)\nabla p - \frac{\partial K(p)}{\partial z} = f, \tag{1}$$

where $p(t)$ is the pressure head at time t, $s(p)$ is water saturation at pressure head p, ρ is water density, ϕ is the porosity of the medium, $K(p)$ the hydraulic conductivity, f represents any water source terms and z is elevation. The equation is then completed by adding boundary and initial conditions. This formulation of the Richards equation is called the *mixed form* References 1 and 3 because the equation is parameterized in p (pressure head) but the time derivative is in terms of s (water saturation). Another formulation of the Richards equation is known as the head-based form, and is popular because the time derivative is written explicitly in terms of the pressure head p.[2,3]

It is important to note that in unsaturated flow both water content, $\phi s(p)$, and hydraulic conductivity, $K(p)$, are functions of the pressure head p. There are several empirical relations[8] used to relate these parameters, including, for example, the Brooks–Corey[9] model and the Van Genuchten model.[2] The Van Genuchten model is slightly more popular because it containts no discontinuities in the functions, unlike the Brooks–Corey model, and therefore avoids the risk of losing the uniqueness of the solution of the semidiscretized equation in space by the well-known Peano Theorem. A version of the Van Genuchten–Richards equation in mixed form model by Celia et al.[3] reported the following choices:

$$s(p) = \frac{\alpha(s_s - s_r)}{\alpha + |p|^{\beta}} + s_r, \tag{2}$$

and

$$K(p) = K_s \frac{a}{a + |p|^{\gamma}}, \tag{3}$$

where the α, β, γ, and a are fitting parameters that are often assumed to be constant in the media; s_r and s_s are the residual and saturated moisture contents, and K_s is the saturated hydraulic conductivity.

Small changes in the pressure head can change the hydraulic conductivity by several orders of magnitude, and $K(p)$ is a highly nonlinear function. The water content curve is also highly nonlinear as saturation can change drastically over a small range of pressure head values. It should be noted that these functions are only valid when the pressure head is negative; that is, the media is semi-saturated; when the media is fully saturated, $K = K_s$, $s(p)$ is equal to the porosity, and the Richards equation reduces to the Darcy linear equation; see the example in Figure 1.

Typical uses of the Richards equation are to simulate infiltration experiments in the laboratory and in the field. These begin with initially dry soil, then water is added to the top of the core sample (or ground surface). These experiments require the simulation of fast changes in pressure head and saturation over the most nonlinear part of the Van Genuchten curves.

2.1 | Cell-centered finite difference discretization

At the beginning of an infiltration experiment, the pressure head p can be close to discontinuous. These large changes are also reflected in the nonlinear terms K and $s(p)$. Taking into account the effect of the initial conditions, the time step can be severely limited if an inappropriate time discretization is chosen. Hydrogeologists are often interested in following

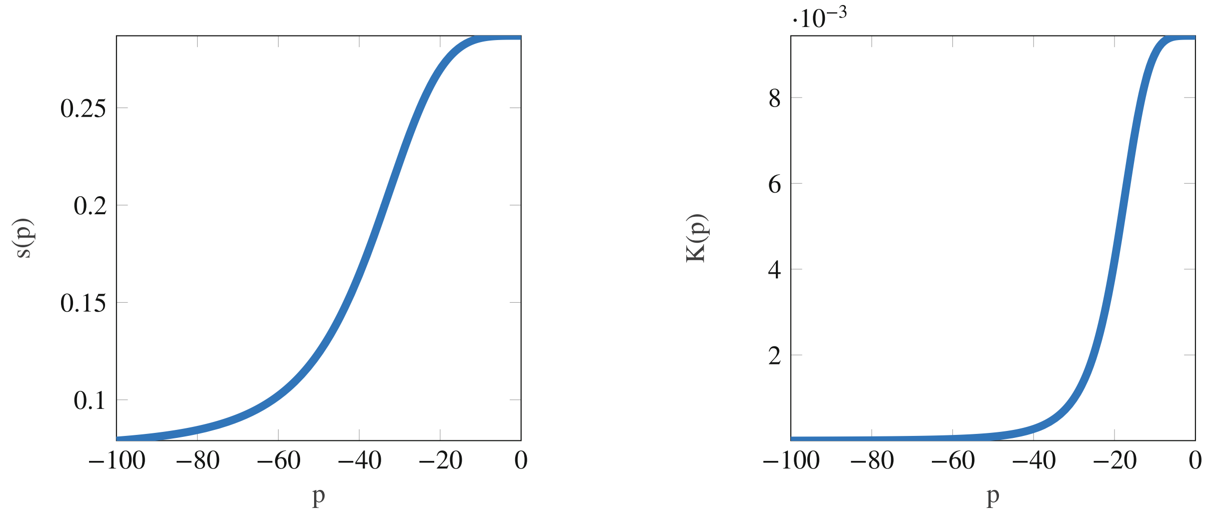


FIGURE 1 Hydraulic conductivity $K(p)$ and volumetric water content $S(p)$ for parameters $\alpha = 1.611\text{e}{+}6$, $\beta = 3.96$, $\gamma = 4.74$, $a = 1.175\text{e}{+}6$, $S_s = 0.287$, $S_r = 0.075$, $K_s = 0.00944$ cm/s.

the process until a steady state is achieved, which may take a long time; therefore, large time steps should be used to avoid excessively expensive simulations. The presence of stiffness and the desire to use long time steps, lead to implicit time integration schemes. In this paper, we consider an implicit backward Euler numerical scheme; higher-order implicit methods are not considered because of the uncertainty associated with the fitting parameters in the Van Genuchten models and the possible low smoothness of s (i.e., the max order of differentiability) can severely reduce the effective order of a (theoretically) high order numerical method.

Successful discretizations in space for Richards equation are based on finite differences;[3,5] using finite elements may require mass lumping in order to recover possible large mass balance errors and undershoot errors ahead of the infiltration front.[3]

We consider here a discretization of the Richards equation on a cell-centered finite difference tensor mesh on a parallelepiped discretized with $\mathbf{N} = (N_x, N_y, N_z)$ nodes, in such a way that the most external nodes are on the physical boundary of the domain. We then denote the cell centers as $\{x_{i,j,k} = (ih_x, jh_y, kh_z)\}_{i,j,k=0}^{N-1}$, for $\mathbf{h} = (h_x, h_y, h_z) = (L_x, L_y, L)/(\mathbf{N} - 1)$, and with the interfaces located at midpoints between adjacent nodes. The time direction is then discretized by considering N_t uniform time steps, that is, the grid $\{t_l = l\Delta t\}_{l=0}^{N_t-1}$ for $\Delta t = 1/(N_t - 1)$; the approximation on the cell (i, j, k) at time step l of the pressure head p is denoted as $p_{i,j,k}^{(l)}$.

With this choice, the discretization of the mixed form of the Richards equation in (1) on the internal nodes of the grid for $l \geq 1$ is written, for $i, j, k = 1, \dots, \mathbf{N} - 2$, as

$$\mathbf{\Phi}_{i,j,k}(\mathbf{p}^{(l)}) = \frac{\rho\phi}{\Delta t}\left(s\left(p_{i,j,k}^{(l)}\right) - s\left(p_{i,j,k}^{(l-1)}\right)\right) + q_{i+1/2,j,k}^{(l)} - q_{i-1/2,j,k}^{(l)} + q_{i,j+1/2,k}^{(l)} - q_{i,j-1/2,k}^{(l)} + q_{i,j,k+1/2}^{(l)} - q_{i,j,k-1/2}^{(l)} + f_{i,j,k} \equiv 0, \tag{4}$$

with

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

in which the generic term ${}^{\mathrm{AV}}K_{L,U}^{(l)}$ represents an average at the interfaces of the function $K(p)$ in (3). The selection of the form of the average term best suited to more realistic simulations does depend on the problem and is still an open issue.[10,11] In particular, the choice is dependent on the type of fluid infiltration one needs to deal with. If we denote by K_U and K_L the two values of the function K on the opposite sides of the interface, for example, $K_U = K(p_{i,j+1,k}^{(l)})$ and $K_L = K(p_{i,j,k}^{(l)})$, the most frequently used formulations are the arithmetic mean,[3] that is, ${}^{\mathrm{ARIT}}K_{L,U}^{(l)} = (K_U + K_L)/2$, the geometric mean,[12] that is, ${}^{GEOM}K^{(l)} = \sqrt{K_U K_L}$, the upstream-weighted mean,[13] that is,

$${}^{\mathrm{UP}}K^{(l)} = \begin{cases} K_U, & p_U - p_L \geq 0, \\ K_L, & p_U - p_L < 0, \end{cases} \tag{5}$$

and the integral mean,[14] that is,

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

It is also possible to employ a combination of the above, using two different means, one for the horizontal and another for the vertical direction, or to use an algorithm that computes the internodal conductivity by using different approaches depending on the terrain and the value of the pressure head.[10]

3 | APPLYING THE INEXACT NEWTON METHOD

We apply at each time step a quasi-Newton method for the solution of the nonlinear system of equations (4) as implemented in the KINSOL parallel library.[6] Let $\mathbf{p}_r$ be the current iterate of pressure head, for each node of the computational mesh and for each time step. A Newton method computes an increment $\mathbf{d}_r$ as the solution of the following equation:

$$J(\mathbf{p}_r)\mathbf{d}_r = -\mathbf{\Phi}(\mathbf{p}_r), \tag{6}$$

where $J(\mathbf{p}_r)$ is the Jacobian matrix of $\mathbf{\Phi}$. Specifically, we consider an inexact Newton solver in which we update the Jacobian matrix as infrequently as possible. This means that a first Jacobian is computed in the initialization phase, that is, in the first step $r = 0$ of the algorithm; then, a new one is built if and only if at least one of the following conditions are met

$$\begin{cases} r \equiv 0 \pmod{10}, \\ |\lambda d_{r-1}|_{D_u,\infty} > 1.5, \\ |\lambda d_r|_{D_u,\infty} < \varepsilon_{\text{machine}}^{2/3} \\ \quad \text{The linear solver failed to converge with the} \\ \quad\quad \text{previous Jacobian (backtracking), or} \\ \quad \text{The line search failed with outdated Jacobian information,} \end{cases} \tag{7}$$

where we are using the scaled norm $|\cdot|_{D_u,\infty} = |D_u \cdot|_\infty$ for a diagonal matrix D_u such that the entries of the scaled vector are almost of the same magnitude when $\mathbf{p}_r$ is close to a solution. Similarly, $|\cdot|_{D_F,\infty} = |D_F \cdot|_\infty$ for a diagonal matrix D_F when we are far from the solution. The value of the step length λ is computed via the Armijo–Goldstein line search strategy, that is, λ is chosen to guarantee a sufficient decrease in $\mathbf{\Phi}$ with respect to the step length as well as a minimum step length to the initial rate of decrease of $\mathbf{\Phi}$.

The Jacobian matrix $J = J_{\mathbf{\Phi}}$ can then be recovered by direct computation from (4) using finite-difference approximations to the derivatives of the constitutive equations in (2), and (3) given by

$$s'(p) = -\frac{\alpha\beta|p|^{\beta-1}\text{sgn}(p)\,(S_s - S_r)}{\left(\alpha + |p|^{\beta}\right)^2}, \quad \text{and} \quad K'(p) = -\frac{a\gamma k_s|p|^{\gamma-1}\text{sgn}(p)}{\left(a + |p|^{\gamma}\right)^2}. \tag{8}$$

At the core of the parallel procedure we tackle the solution of the (right preconditioned) linear system

$$JM^{-1}(M\mathbf{d}_r) = -\mathbf{\Phi}(\mathbf{p}_r), \tag{9}$$

where J could be either a freshly computed Jacobian for the vector $\mathbf{p}_r$, or the Jacobian coming from a previous step. The linear iterative solver for the Newton equations should handle the tradeoff between using a tolerance that is large enough to avoid oversolving, and reducing the number of iterations to attain the prescribed convergence. In the following, we focus on a choice for the preconditioner M in (9) which allows to balance accuracy and efficiency on parallel distributed-memory architectures when very large scale simulations have to be carried out.

In Section 4 we investigate the asymptotic spectral properties of the sequence of the Jacobian matrices. This information will then be used to devise an *asymptotically spectrally equivalent* sequence of symmetric and positive definite matrices. Then, to approximate the action of the inverses of the approximating sequence, we will employ some preconditioners from the PSCToolkit framework described in Section 5. The aim to experiment the functionalities of PSCToolkit for building and apply preconditioners for the linear systems arising in the quasi-Newton procedure by KINSOL, we developed some KINSOL modules that enable it to use the solvers and preconditioners from PSCToolkit inside its Newton-based nonlinear procedure. These interfaces are written in C from the KINSOL library and use the C/Fortran2003 interfaces from the PSCToolkit libraries, guaranteeing a full interoperability of the data structures, that is, they do not require producing any auxiliary copy of KINSOL objects for translation into PSCToolkit objects; everything can be manipulated from KINSOL directly into the native formats for PSCToolkit. The details about the implementation of the relevant APIs, and the operators made available by the interfacing are described in the documentation for the interface that can be downloaded from https://github.com/Cirdans-Home/kinsol-psblas. This interfacing had the twofold aim of extending the PSCToolkit to handle nonlinear algebraic equations, as well as to extend the KINSOL library with new methods for solving sparse linear systems on high-end supercomputers.

4 | SPECTRAL ANALYSIS OF THE JACOBIAN SEQUENCE

In general, the construction of the preconditioner M in (9) depends on the choice of the average for the discretization used in (4). To formulate a proposal for M, we divide the discussion into two steps: first, we look for a suitable matrix M to precondition the Jacobian matrix J associated with the different averages; then we discuss how we can efficiently setup and apply M^{-1} inside the Krylov subspace method on a high-end parallel computer.

4.1 | Tools for the spectral analysis

Our idea to compute M starts by investigating the distribution of the eigenvalues, $\{\lambda_j(J_{\mathbf{N}})\}_{\mathbf{j}=\mathbf{1}}^{\mathbf{N}=(N_x,N_y,N_z)}$, for the Jacobian matrix $J_{\mathbf{N}}$ of size $N = N_xN_yN_z$. Specifically, we look for a measurable function $f : D \subset \mathbb{R}^k \to \mathbb{C}$ to associate to the sequence $\{J_{\mathbf{N}}\}_{\mathbf{N}}$ for which we can prove the following *asymptotic* relation

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

where $\mu_k(\cdot)$ is the Lebesgue measure on $\mathbb{R}^k$, $0 < \mu_k(D) < \infty$, and $C_c(\mathbb{C})$ is the space of continuous functions with compact support. Despite the apparently technical form of the previous expression, we can easily summarize the information contained in f. 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, that is, the function f, that we will call *symbol* of the sequence of matrices $\{J_{\mathbf{N}}\}_{\mathbf{N}}$, provides an accurate description of their spectrum asymptotically.

In order to achieve this result, we need to introduce some preliminary tools. To simplify notation, let us start by focusing on the one dimensional problem to select only an expression for ${}^{\mathrm{AV}}K_{L,U}$ related to the flux oriented toward the z axis. For these cases, we are interested in the overall behavior for both the eigenvalues and the singular values of the Jacobian matrices. Formally, we are interested in the computation of the so-called *singular* and *spectral* value symbol for the sequence of the Jacobians.

Definition 1. Let $\{A_N\}_N$ be a sequence of matrices and let $f : D \subset \mathbb{R}^k \to \mathbb{C}$ be a measurable function defined on a set D with $0 < \mu_k(D) < \infty$.

- $\{A_N\}_N$ has a singular value distribution described by f, and we write $\{A_N\}_N \sim_\sigma f$, if

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} F(\sigma_i(A_N)) = \frac{1}{\mu_k(D)}\int_D F(|f(\mathbf{x})|)\mathrm{d}\mathbf{x}, \qquad \forall\, F \in C_c(\mathbb{R}).$$

  In this case, f is called the *singular value symbol* of $\{A_N\}_N$.
- $\{A_N\}_N$ has a spectral (or eigenvalue) distribution described by f, and we write $\{A_N\}_N \sim_\lambda f$, if

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

  In this case, f is called the *spectral (or eigenvalue) symbol* of $\{A_N\}_N$.

If $\{A_N\}_N$ has both a singular value and a spectral distribution described by f, we write $\{A_N\}_N \sim_{\sigma,\lambda} f$.

Moreover, we refer to a sequence of matrices $\{Z_N\}_N$ such that $\{Z_N\}_N \sim_\sigma 0$ as a zero-distributed sequence. Examples of sequences for which we can easily compute such symbols are the nth diagonal sampling matrix generated by $a : [0,1] \to \mathbb{C}$ and $N \in \mathbb{N}$, that is the $N \times N$ diagonal matrix given by

$$D_N(a) = \mathrm{diag}_{i=1,\dots,N}\, a\left(\frac{i}{N}\right),$$

and the Toeplitz sequences, that is, for a $N \in \mathbb{N}$ and $f : [-\pi,\pi] \to \mathbb{C}$ a $L^1([-\pi,\pi])$ function, the $N \times N$ matrix

$$T_N(f) = [f_{i-j}]_{i,j=1}^N,$$

where the numbers f_k are the Fourier coefficients of f,

$$f_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(\theta)\mathrm{e}^{-\mathrm{i}\,k\theta}\mathrm{d}\theta, \qquad k \in \mathbb{Z}.$$

For these sequences, if $f \in L^1([-\pi,\pi])$ then $\{T_N(f)\}_N \sim_\sigma f$, while if $f \in L^1([-\pi,\pi])$ and f is real then $\{T_N(f)\}_N \sim_\lambda f$. A similar relationship holds for the case of multilevel Toeplitz matrices.

To compute the asymptotic spectral/singular value distribution of more general matrix sequences we need to expand our set of tools, and for this task, we use the Generalized Locally Toeplitz (GLT) sequences.[15,16] GLT sequences are sequences of matrices equipped with a measurable function $\kappa : [0,1] \times [-\pi,\pi] \to \mathbb{C}$ called *symbol*; we will use the notation $\{A_N\}_N \sim_{\mathrm{GLT}} \kappa$ to indicate that $\{A_N\}_N$ is a GLT sequence with symbol κ. We can characterize the sequences by the following list of properties.

GLT 1. If $\{A_N\}_N \sim_{\mathrm{GLT}} \kappa$ then $\{A_N\}_N \sim_\sigma \kappa$. If $\{A_N\}_N \sim_{\mathrm{GLT}} \kappa$ and the matrices A_N are Hermitian then $\{A_N\}_N \sim_\lambda \kappa$.

GLT 2. If $\{A_N\}_N \sim_{\mathrm{GLT}} \kappa$ and $A_N = X_N + Y_N$, where

- every X_N is Hermitian,
- $\|X_N\|, \|Y_N\| \leq C$ for some constant C independent of N, and $\|\cdot\|$ the spectral norm, that is, the norm induced by the Euclidean vector norm,
- $N^{-1}\|Y_N\|_1 \to 0$, and $\|\cdot\|_1$ the Schatten–1 norm, that is, the sum of the singular values.

then $\{A_N\}_N \sim_\lambda \kappa$.

GLT 3. We have

- $\{T_N(f)\}_N \sim_{\mathrm{GLT}} \kappa(x,\theta) = f(\theta)$ if $f \in L^1([-\pi,\pi])$,
- $\{D_N(a)\}_N \sim_{\mathrm{GLT}} \kappa(x,\theta) = a(x)$ if $a : [0,1] \to \mathbb{C}$ is Riemann-integrable,
- $\{Z_N\}_N \sim_{\mathrm{GLT}} \kappa(x,\theta) = 0$ if and only if $\{Z_N\}_N \sim_\sigma 0$.

GLT 4. If $\{A_N\}_N \sim_{\mathrm{GLT}} \kappa$ and $\{B_N\}_N \sim_{\mathrm{GLT}} \xi$ then

- $\{A_N^*\}_N \sim_{\mathrm{GLT}} \overline{\kappa}$,
- $\{\alpha A_N + \beta B_N\}_N \sim_{\mathrm{GLT}} \alpha\kappa + \beta\xi$ for all $\alpha, \beta \in \mathbb{C}$,
- $\{A_N B_N\}_N \sim_{\mathrm{GLT}} \kappa\xi$.

GLT 5. If $\{A_N\}_N \sim_{\mathrm{GLT}} \kappa$ and $\kappa \neq 0$ a.e. then $\{A_N^\dagger\}_N \sim_{\mathrm{GLT}} \kappa^{-1}$.

We call *unilevel* sequences those for which the dimension is characterized by a single index N, we call d-level those in which the dimension is characterized by a multi-index $\mathbf{N} \in \mathbb{N}^d$. All definitions and the related GLT tools have been generalized to the multi-level setting.[16]

Another tool we will use in the proofs is the modulus of continuity for a function g. We report here the definition for completeness.

Definition 2 (modulus of continuity). If $g : D \to \mathbb{R}$ is continuous over $D \subseteq \mathbb{R}^k$ for some k, we denote by $\omega_g(\cdot)$ the *modulus of continuity* of g,

$$\omega_g(h) = \sup_{\substack{\mathbf{x},\mathbf{y}\in D \\ \|\mathbf{x}-\mathbf{y}\|\leq h}} |g(\mathbf{x}) - g(\mathbf{y})|, \qquad h > 0.$$

Furthermore, we recall also that by the Heine–Cantor theorem every continuous function on a compact set is uniformly continuous, therefore for a continuous function on a compact set any modulus of continuity is required to be infinitesimal at 0.

Remark 1. In all the following analysis, to simplify the notation, we remove from the vector $\mathbf{p}^{(k,l+1)}$ the index k denoting the dependence on the iterate of the Newton method. The spectral analysis uses only the entries $p_i^{(l+1)}$ and does not depend on the step at which such a vector has been obtained.

4.2 | Arithmetic average

The simplest choice for the ${}^{\mathrm{AV}}K_{L,U}$ is given by the arithmetic mean of the values of K in (3) at the two sides of the interface, that is,

$$
\begin{aligned}
{}^{\mathrm{ARIT}}K_{i+1,i}^{(l+1)} &= \frac{1}{2}\left(K(p_i^{(l+1)}) + K(p_{i+1}^{(l+1)})\right),\\
{}^{\mathrm{ARIT}}K_{i-1,i}^{(l+1)} &= \frac{1}{2}\left(K(p_i^{(l+1)}) + K(p_{i-1}^{(l+1)})\right).
\end{aligned}
$$

With this choice, Equation (4) becomes

$$
\begin{aligned}
{}^{\mathrm{ARIT}}\Phi_i(\mathbf{p}^{(l+1)}) = &\frac{S\left(p_i^{(l+1)}\right) - S\left(p_i^{(l)}\right)}{\Delta t}\\
&- \frac{1}{2h_z^2}\left[\left(p_{i+1}^{(l+1)} - p_i^{(l+1)}\right)\left(K\left[p_{i+1}^{(l+1)}\right] + K\left[p_i^{(l+1)}\right]\right)\right.\\
&\left. - \left(p_i^{(l+1)} - p_{i-1}^{(l+1)}\right)\left(K\left[p_{i-1}^{(l+1)}\right] + K\left[p_i^{(l+1)}\right]\right)\right]\\
&- \frac{1}{2h_z}\left(K\left[p_{i+1}^{(l+1)}\right] - K\left[p_{i-1}^{(l+1)}\right]\right)
\end{aligned}
$$

Then, the Jacobian matrix is, as in all the other one dimensional cases, a tridiagonal matrix with entries

$$
{}^{\mathrm{ARIT}}J_N = \mathrm{tridiag}(\eta_i, \zeta_i, \xi_i) = \begin{bmatrix} \zeta_1 & \xi_1 & & \\ \eta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \xi_{N-3} \\ \eta_{N-2} & \zeta_{N-2} & & \end{bmatrix}, \tag{10}
$$

where,

$$
\begin{aligned}
\zeta_i = &\frac{\Delta t K\left[p_{i-1}^{(l+1)}\right] + \Delta t K\left[p_{i+1}^{(l+1)}\right] + 2\Delta t K\left[p_i^{(l+1)}\right] + 2h_z^2 S'\left(p_i^{(l+1)}\right)}{2\Delta t h_z^2}\\
&- \frac{p_{i-1}^{(l+1)}K'\left(p_i^{(l+1)}\right)}{2h_z^2} + \frac{p_i^{(l+1)}K'\left(p_i^{(l+1)}\right)}{h_z^2} - \frac{p_{i+1}^{(l+1)}K'\left(p_i^{(l+1)}\right)}{2h_z^2},\\
\xi_i = &-\frac{K\left[p_{i+1}^{(l+1)}\right] + K\left[p_i^{(l+1)}\right] + h_z K'\left(p_{i+1}^{(l+1)}\right)}{2h_z^2} - \frac{p_{i+1}^{(l+1)}K'\left(p_{i+1}^{(l+1)}\right)}{2h_z^2}\\
&+ \frac{p_i^{(l+1)}K'\left(p_{i+1}^{(l+1)}\right)}{2h_z^2},\\
\eta_i = &-\frac{K\left[p_{i-1}^{(l+1)}\right] + K\left[p_i^{(l+1)}\right] - h_z K'\left(p_{i-1}^{(l+1)}\right)}{2h_z^2} - \frac{p_{i-1}^{(l+1)}K'\left(p_{i-1}^{(l+1)}\right)}{2h_z^2}\\
&+ \frac{p_i^{(l+1)}K'\left(p_{i-1}^{(l+1)}\right)}{2h_z^2}.
\end{aligned}
$$

To perform the spectral analysis we first rewrite the Jacobian matrices as a sum of matrices that are easier to investigate, we use then the $*$-algebra (**GLT4**) and perturbation techniques (**GLT2**) to obtain the spectral information we look for. In particular, to analyze the Jacobian in (10) we separate it into three parts, one for the time-stepping, one taking into account the Darcian diffusion, and the last one related to the transport along the z-axis, that is,

$$\left\{{}^{\mathrm{ARIT}}J_N\right\}_N = \left\{{}^{\mathrm{ARIT}}D_N\right\}_N + \left\{{}^{\mathrm{ARIT}}L_N\right\}_N + \left\{{}^{\mathrm{ARIT}}T_N\right\},$$

where $\left\{{}^{\mathrm{ARIT}}D_N\right\}_N$ is the scaled diagonal matrix sampling $s'(p)$ on the function p approximated at the $(i+1)$th time step, that is,

$$ {}^{\mathrm{ARIT}}D_N = \frac{1}{\Delta t}\mathrm{diag}\left(s'\left[p_i^{(l+1)}\right]\right)_{i=1}^{N-2}. \tag{11}$$

We can express the Darcian diffusion part as

$$\begin{aligned}
{}^{\mathrm{ARIT}}L_N = \frac{1}{2h_z^2}&\mathrm{tridiag}\left(-K\left[p_{i-1}^{(l+1)}\right], 2K\left[p_i^{(l+1)}\right], -K\left[p_{i+1}^{(l+1)}\right]\right)_{i=1}^{N-2}\\
&+\frac{1}{2h_z^2}\mathrm{tridiag}\left(0, K\left[p_{i-1}^{(l+1)}\right], -K\left[p_i^{(l+1)}\right]\right)_{i=1}^{N-2}\\
&+\frac{1}{2h_z^2}\mathrm{tridiag}\left(-K\left[p_i^{(l+1)}\right], K\left[p_{i+1}^{(l+1)}\right], 0\right)_{i=1}^{N-2}\\
&+\frac{1}{2h_z^2}\mathrm{tridiag}\left(-p_{i-1}^{(l+1)}K'\left[p_{i-1}^{(l+1)}\right], 2p_i^{(l+1)}K'\left[p_i^{(l+1)}\right], -p_{i+1}^{(l+1)}K'\left[p_{i+1}^{(l+1)}\right]\right)_{i=1}^{N-2}\\
&+\frac{1}{2h_z^2}\mathrm{tridiag}\left(0, -p_{i-1}^{(l+1)}K'\left[p_i^{(l+1)}\right], p_i^{(l+1)}K'\left[p_{i+1}^{(l+1)}\right]\right)_{i=1}^{N-2}\\
&+\frac{1}{2h_z^2}\mathrm{tridiag}\left(p_i^{(l+1)}K'\left[p_{i-1}^{(l+1)}\right], -p_{i+1}^{(l+1)}K'\left[p_i^{(l+1)}\right], 0\right)_{i=1}^{N-2},
\end{aligned}$$

while the remaining part is given by

$$ {}^{\mathrm{ARIT}}T_N = -\frac{1}{2h_z}\mathrm{tridiag}\left(-K'\left[p_{i-1}^{(l+1)}\right], 0, K'\left[p_{i+1}^{(l+1)}\right]\right)_{i=1}^{N-2}. \tag{12}$$

Lemma 1. *Assuming that $C = \lim_{N,N_t\to+\infty} h_z^2/\Delta t$ is a finite nonzero constant, then the matrix sequence $\{h_z^{2\,ARIT}D_N\}_N$ has GLT symbol $Cs'(p(\psi(x)))$, for s' in (8), for $\psi(x)$ the function mapping the $[0,1]$ interval to the domain of definition for the z variable.*

Proof. We observe that ${}^{\mathrm{ARIT}}D_N$ is the $(N-2)$th diagonal sampling matrix for the function $s'(p(z))$. Therefore, it is one of the sequences of which we know the distribution (**GLT3**), and we conclude that

$$\left\{h_z^2\ {}^{\mathrm{ARIT}}D_N\right\}_N \sim_{\mathrm{GLT}} Cs'(p(\psi(x))).$$

∎

Remark 2. We observe that the assumption on the ratio C in the above Lemma 1 is such that

$$\exists C > 0\ :\ \frac{h_z^2}{\Delta t} \sim C, \qquad \text{for } h_z, \Delta t \to 0, \quad (N, N_t \to +\infty),$$

that is, it is bounded whenever we impose the compatibility conditions for the relation between the time and space discretization. Therefore the assumption can be justified in terms of the analysis of the error and the physics of the problem.

Lemma 2. *The matrix sequence $\left\{h_z^{2\ ARIT}L_N\right\}_N$ has GLT symbol*

$$K(p(\psi(x)))(2-2\cos(\theta)),$$

where $\psi(x)$ is the function mapping $[0,1]$ interval to the domain of definition for the z variable.

Proof. Let us start from the first tridiagonal matrix

$$A_N = \frac{1}{2}\text{tridiag}\left(-K\left[p_{i-1}^{(l+1)}\right], 2K\left[p_i^{(l+1)}\right], -K\left[p_{i+1}^{(l+1)}\right]\right)_{i=1}^{N-2},$$

to produce its GLT symbol, we can consider the matrix sequence

$$D_N\left(K\left(p^{(l+1)}\right)\right)T_N(2-2\cos\theta).$$

If we then perform a direct comparison of

$$A_N - \frac{1}{2}D_N\left(K\left(p^{(l+1)}\right)\right)T_N(2-2\cos\theta),$$

we observe that the only nonzero entries are the ones on the lower and upper diagonals, in which we have the differences $K\left(p_{i\pm1}^{(l+1)}\right) - K\left(p_i^{(l+1)}\right)$ on the grid points $|z_{i\pm1} - z_i| = h_z$. From this, we bound the modulus of each off-diagonal entries of

$$A_N - \frac{1}{2}D_N\left(K\left(p^{(l+1)}\right)\right)T_N(2-2\cos\theta),$$

by the modulus of continuity of the function $K(p(z))$ that is expressed by $\frac{1}{2}\omega_{K(p(z))}(h_z)$ as in Definition 2 and then the 1-norm and the ∞-norm of the difference $A_N - \frac{1}{2}D_N\left(K\left(p^{(l+1)}\right)\right)T_N(2-2\cos\theta)$ are bounded by $\omega_{K(p(z))}(h_z)$. Thus,

$$\|A_N - \frac{1}{2}D_N\left(K\left(p^{(l+1)}\right)\right)T_N(2-2\cos\theta)\| \le \omega_{K(p(z))}(h_z) \to 0 \text{ for } N \to +\infty,$$

therefore, $Z_N = A_N - \frac{1}{2}D_N\left(K\left(p^{(l+1)}\right)\right)T_N(2-2\cos\theta)$ is distributed as zero. Then, a direct application of **GLT3** and **GLT4** tells us that $A_N \sim_{\text{GLT}} 1/2K(p(\psi(x)))(2-2\cos(\theta))$. With minor modifications, the same arguments can be applied to the other parts of $\{h_z^{2\,\text{ARIT}}L_N\}_N$, and by means of the *-algebra properties from **GLT4**, we can write

$$\begin{aligned}
\left\{h_z^{2\text{ARIT}}L_N\right\}_N \sim_{\text{GLT}} & \frac{1}{2}K(p(\psi(x)))(2-2\cos(\theta)) \\
& + \frac{1}{2}K(p(\psi(x)))(2-2\cos(\theta))(1-e^{-i\theta}) \\
& + \frac{1}{2}K(p(\psi(x)))(2-2\cos(\theta))(1-e^{-i\theta}) \\
& + \frac{1}{2}p(\psi(x))K'(p(\psi(x)))(2-2\cos(\theta)) \\
& + \frac{1}{2}p(\psi(x))K'(p(\psi(x)))(-1+e^{i\theta}) \\
& + \frac{1}{2}p(\psi(x))K'(p(\psi(x)))(-1+e^{i\theta}) \\
= & K(p(\psi(x)))(2-2\cos(\theta)).
\end{aligned}$$

■

Remark 3. In the computation of the GLT symbol of the sequence ${}^{\text{ARIT}}L_N$ we have that a part of the distribution simplifies itself. This appears as if there was a simplification of the lower order terms induced by the arithmetic mean, that is, in a chain-rule style the correction due to the term $pK'(p)$ cancels out.

Lemma 3. *The matrix sequence* $\left\{h_z^{2\,\text{ARIT}}T_N\right\}_N$ *is a sequence distributed as zero.*

Proof. By construction, the matrices of the sequence $\left\{h_z^{2\,\text{ARIT}}T_N\right\}_N$ are such that

$$\|h_z^{2\;\text{ARIT}}T_N\| \le h_z\|K'(p)\|_\infty \le \frac{C}{N},$$

for some constant C independent of N. Therefore, $\left\{h_z^{2\,\text{ARIT}}T_N\right\}_N \sim_\sigma 0$. ■

Theorem 1. *The matrix sequence $\left\{h_z^2\,{}^{ARIT}J_N\right\}_N$ is distributed in the eigenvalue sense as the function*

$$\kappa(x,\theta) = Cs'(p(\psi(x))) + K(p(\psi(x)))(2-2\cos(\theta)),$$

where $\psi(x)$ is the function mapping $[0,1]$ interval to the domain of definition for the x variable.

Proof. The proof follows in two steps from applying **GLT4** with the symbols obtained in Lemmas 1,2, and 3, we first find that $\left\{h_z^2\,{}^{\text{ARIT}}J_N\right\}_N \sim_{\text{GLT}} \kappa(z,\theta) = Cs'(p(\psi(x))) + K(p(\psi(x)))(2-2\cos(\theta))$. Then, by **GLT2**, that holds in virtue of Lemma 3, we have that the distribution holds also in the eigenvalue sense, since we can write

$$h_z^2 J_N = h_z^2 D_N + h_z^2 L_N + h_z^2 T_N = X_N + Y_N \;\text{ with }\; X_N = h_z^2 D_N + h_z^2 T_N = X_N^T, \quad \forall\, N$$

furthermore, by direct inspection

$$\|X_N\| \leq 4(\|K(p)\|_\infty + \|pK(p)\|_\infty) + C\|s'(p)\|_\infty = C_1,$$

independent of N, and, due to the fact that the we can bound the spectral norm with the p-norms

$$\|X\| \leq \sqrt{|X|_1 |X|_\infty},$$

and analogously $\|Y_N\| \leq h_z\|K'\|_\infty \to 0$ for $h_z \to 0$. Furthermore, the Schatten 1-norm $\|\cdot\|_1$ can be always estimated in terms of the spectral norm as

$$\|X\|_1 \leq \text{rank}(X)\|X\| \leq m\|X\|, \qquad \forall X \in \mathbb{C}^{m\times m},$$

thus by the inequality in the proof of Lemma 3, we conclude that $\|Y_N\|_1 = O(1)$, hence **GLT2**. ∎

Remark 4. From Theorem 1 we infer that the ill-conditioning in the Jacobian matrices is determined by two main factors. On the one hand, we have the ill-conditioning due to the diffusion operator that drives the eigenvalues to zero as an $O(h_z^2)$. On the other, we observe that this effect is enhanced by the behavior of the hydraulic conductivity $K(p)$ at the current time step/Newton iterate. Indeed, for lower values of the pressure head, this further enhances the decay to zero of the eigenvalues. This implies also that, when the soil becomes more saturated, the ill-conditioning is due only to diffusion.

Now, with the choice of this average for all the spatial dimensions, we can explicitly write the nonzero elements of the Jacobian matrix by maintaining the local (i,j,k)-indices, that is, in a way that is independent of the selected ordering; see the Supplementary Material Section B for the complete expression. To complete the discretization we only need to impose and discretize the boundary conditions. For the test case considered here, we focus on Dirichlet boundary conditions that can be either homogeneous, to identify water table boundary conditions, or time dependent.

Theorem 2. *Assuming that $\mathbf{h} = \nu(q_1,q_2,q_3)$, $q_i \in \mathbb{Q}^+ = \{q \in \mathbb{Q} \;:\; q > 0\}$, and that $C = \lim_{\nu,\Delta t\to 0} \frac{\nu^2}{\Delta t}$ is finite and non zero then the sequence $\left\{J_{\mathbf{N}}^{(k,j)}\right\}_{\mathbf{N}}$ obtained with the entries in (B1), for $K(p)$, $s(p)$ in (2), (3) is distributed in the sense of the eigenvalues as the function*

$$\begin{aligned} f(\mathbf{x},\theta) = C\rho\phi s'(\mathbf{p}^{(k,j)}(\psi(\mathbf{x}))) \qquad & \mathbf{x} \in [0,1]^3,\; \boldsymbol{\theta} \in [-\pi,\pi]^3, \\ + K(\mathbf{p}^{(k,j)}(\psi(\mathbf{x})))(q_1^{-2}(2-2\cos(\theta_1)) &+ q_2^{-2}(2-2\cos(\theta_2)) + q_3^{-2}(2-2\cos(\theta_3))), \end{aligned}$$

where $\psi(\mathbf{x})$ is the function mapping the $[0,1]^3$ cube to the physical domain.

Proof of Theorem 2. Moving from the results in Theorem 1 to the ones in Theorem 2 is just a technical rewriting. The proofs in Lemma 1, and Lemma 3 remain the same. We need to rewrite the decomposition of ${}^{\text{ARIT}}L_N$ exploiting the fact that the tensor structure in the grid corresponds to a Kronecker structure in the matrix. ∎

We can prove the exact same statement given in Theorem 2 also for the upstream mean together with the expression of the complete Jacobian. By virtue of the mechanism applied, the proof is completely analogous and requires only some technical adjustments related to the corrections in rank. The details and full Jacobian expression for this case are given in the Supplementary Material, see Sections C and D.

Theorem 3. *Assuming that* $\mathbf{h} = v(q_1, q_2, q_3)$, $q_i \in \mathbb{Q}^+ = \{q \in \mathbb{Q} \ : \ q > 0\}$, *and that* $C = \lim_{v,\Delta t \to 0} \frac{v^2}{\Delta t}$ *is finite and non zero then the sequence* $\left\{ J_{\mathbf{N}}^{(k,j)} \right\}_{\mathbf{N}}$ *obtained with the choice of the upstream average (entries in (C2)) or with the arithmetic average (entries in (B1)), for* $K(p)$, $s(p)$ *in (2), (3) is distributed in the sense of the eigenvalues as the function*

$$\begin{aligned} f(\mathbf{x}, \theta) = C\rho\phi s'(\mathbf{p}^{(k,j)}(\psi(\mathbf{x}))) \qquad & \mathbf{x} \in [0,1]^3, \ \boldsymbol{\theta} \in [-\pi, \pi]^3, \\ + K(\mathbf{p}^{(k,j)}(\psi(\mathbf{x})))(q_1^{-2}(2 - 2\cos(\theta_1)) + q_2^{-2}(2 - 2\cos(\theta_2)) &+ q_3^{-2}(2 - 2\cos(\theta_3))), \end{aligned}$$

where $\psi(\mathbf{x})$ *is the function mapping the cube* $[0,1]^3$ *to the physical domain.*

We can conclude that, independently of the mean used, the asymptotic behavior of the spectrum remains the same and it is dominated by the diffusion operator. This suggests the idea of how to define an appropriate sequence of preconditioners for the underlying problem. Specifically, we use the matrix sequence generated by the considered discretization of the Jacobians of (4) having modified the flux term q to be:

$$q_{\text{prec.}} = -K(p)\nabla p.$$

In practice, we disregard all the terms related to the fluid motion along the $\hat{z}$ axis thus using the Jacobians of the nonlinear diffusion operators. using the Jacobian matrices of the nonlinear diffusion operators only. Indeed, these diffusion matrices have the same spectral distribution we have proved with Theorem 3, and therefore by **GLT4** and **GLT5** they provide an optimal preconditioner for the underlying sequence of matrices. We note that, to the best of our knowledge, this is the first theoretical justification of a very common approach in preconditioning Newton correction linear systems for solving the Richards equation.[5,17,18] We are now left with the problem of inverting the asymptotically spectrally equivalent sequence we have just built from the nonlinear diffusion. To face this task, we will perform a further approximation of the theoretical operator's sequence by employing the preconditioners library available in PSCToolkit, a software framework recently proposed for scalable simulations on high-end supercomputers.[*] We point out that the inversions of the theoretical preconditioners will be based on Algebraic MultiGrid methods, as explained in Section 5.2, whose convergence properties for spd matrices, such as those arising from diffusion operators, are well-known and are widely discussed in the literature.[7,19]

5 | SOFTWARE FRAMEWORK FOR VERY LARGE-SCALE SIMULATIONS

In this work, we employ the recently proposed PSCToolkit software framework for parallel sparse computations on current petascale supercomputers. PSCToolkit is composed of two main libraries, named PSBLAS (Parallel Sparse Basic Linear Algebra Subprograms),[20,21] and AMG4PSBLAS (Algebraic MultiGrid Preconditioners for PSBLAS).[7] PSBLAS implements all the main computational building blocks for iterative Krylov subspace linear solvers on parallel computers made of multiple nodes; a plugin for NVIDIA GPUs allows the exploitation of these devices in main sparse matrix and vector computations on hybrid architectures. The toolkit also implements a number of support functionalities to handle the construction of sparse matrices and of their communication structure.

AMG4PSBLAS is a package of preconditioners leveraging on PSBLAS kernels and providing one-level Additive Schwarz (AS) and Algebraic MultiGrid (AMG) preconditioners for PSBLAS-based Krylov solvers. For the present work, we developed a new set of interfaces allowing seamless integration of linear solvers and preconditioners from PSBLAS and AMG4PSBLAS into the SUNDIALS/KINSOL library,[6] in order to use the KINSOL version of the modified Newton methods to solve the discretized Richards nonlinear equations. Interfacing with KINSOL provides a double advantage: the extension of PSCToolkit to handle nonlinear algebraic equations, as well as the extension of the KINSOL library with new methods for solving sparse linear systems on high-end supercomputers.

In the following we describe the details of the proposed parallel Richards solver implemented in C by using PSCToolkit facilities for data generation and distribution and for preconditioner setup and application inside Krylov solvers, and on top of KINSOL facilities for the quasi-Newton procedure as described in Section 3.

5.1 | Domain decomposition and data partitioning

In our parallel solution procedure of the discrete problem in (4), we employ a 2D block decomposition of the parallelepipedal domain Ω of size $[0, L_x] \times [0, L_y] \times [0, L]$, that is, we partition only in the horizontal directions. Indeed, in realistic simulations, one often needs to work with a wide horizontal domain, with a fixed size and resolution for the vertical layer and the vertical physical fields. The above domain decomposition corresponds to a mesh partitioning in which each parallel process owns all degrees of freedom (dofs) in the z direction, while uniform partitioning is applied in the x and y directions. PSBLAS provides all the functionalities needed for very general parallel mesh partitioning and to set up parallel data structures by compact and easy-to-use interfaces. All PSBLAS routines have a pure algebraic interface, where the main data structures are a *distributed sparse matrix* and a corresponding *communication descriptor*. The procedure for mesh partitioning builds a local sparse matrix of possibly noncontiguous rows of the global matrix, where each row corresponds to a local mesh dof and is handled through a local numbering scheme, and automatically builds an index map object contained in the communication descriptor to keep track of the correspondence between local and global indices. The communication descriptor also includes information needed for data communication among processes which is automatically generated after the mesh partitioning procedure and handled internally by the software.

5.2 | AMG4PSBLAS preconditioners

The results in Theorem 2 suggest that we can expect to achieve good convergence in the iterative solution of the linear systems of the quasi-Newton steps (6) by using a matrix M with symbol given by $f(\mathbf{x}, \boldsymbol{\theta})$ as a preconditioner. By means of the $*$-algebra properties in **GLT4** and **GLT5**, this would guarantee a sequence $\{M^{-1} J_{\boldsymbol{\Phi}}(\mathbf{p}^{(l)})\}_{\mathbf{N}} \sim_{\lambda} 1$. Informally, by the spectral symbol, this implies that we can expect the spectrum of the sequence of the preconditioned matrices to have a cluster of eigenvalues in 1. While this would guarantee the theoretical properties we look for, we also need to approximate the inverse of the symmetric and positive definite (*spd* for short) preconditioner. To this aim, we exploit some of the methods available in AMG4PSBLAS.[7]

AMG4PSBLAS includes iterative methods for the approximation of the inverse of the preconditioner M, including domain decomposition techniques of Additive Schwarz (AS) type and AMG methods based on aggregation of unknowns; details on the methods and their parallel versions are available in the literature.[7,22,23] In the following we describe the main features of the preconditioners selected for the experiments discussed in section 6.

In the AS methods, the index space, that is, the set of row/column indices of M, $\Omega^N = \{1, 2, \dots, N\}$, is divided into m, possibly overlapping, subsets Ω_i^N of size N_i. For each subset, we can define the restriction operator R_i which maps a vector $x \in \mathcal{R}^N$ to the vector $x_i \in \mathcal{R}^{N_i}$ made of the components of x having indices in Ω_i^N, and the corresponding prolongation operator $P_i = (R_i)^T$. The restriction of M to the subspace Ω_i^N is then defined by the Galerkin product $M_i = R_i M P_i$. The Additive Schwarz preconditioner for M is defined as the following matrix:

$$M_{AS}^{-1} = \sum_{i=1}^{m} P_i (M_i)^{-1} R_i, \tag{13}$$

where M_i is supposed to be nonsingular and an inverse (or an approximation of it) can be computed by an efficient algorithm. We observe that in the case of nonoverlapping subsets Ω_i^N, the AS preconditioner reduces to the well-known block-Jacobi preconditioner.

AMG4PSBLAS includes all the functionalities for setup and application of the preconditioner in (13) in a parallel setting, where the original index set Ω has been partitioned into subsets Ω_i, each of which is owned by one of the m parallel processes so that the inverse of M_i can be locally computed by a LU factorization or variants of incomplete LU factorizations and sparse approximate inverses.

AS methods can also be applied as smoothers in an AMG procedure, which is the main focus of the AMG4PSBLAS software framework. In this case, the inverse of the preconditioner matrix is defined by the recursion below. Let M_l be a sequence of spd coarse matrices obtained from M by a Galerkin product $M_{l+1} = (P_l)^T M_l P_l,\ l = 0, \dots, n_l - 1$, with $M_0 = M$, and P_l a sequence of prolongation matrices of size $N_l \times N_{l+1}$, with $N_0 = N$ and $N_{l+1} < N_l$. Let S_l be a convergent smoother with respect to the M_l−inner product, the well-known symmetric V-cycle, with one smoother iteration applied at each level before and after the coarse-level correction, defines the inverse of the preconditioner matrix as a sequence of the following matrices:

$$\overline{M}_l = (S_l)^{-T} + (S_l)^{-1} - (S_l)^{-T} M_l (S_l)^{-1} + (I - (S_l)^{-T} M_l)(P_l \overline{M}_{l+1} (P_l)^T)(I - M_l (S_l)^{-1}) \quad \forall l,$$

assuming that $\overline{M}_{n_l} \approx (M_{n_l})^{-1}$ is an approximation of the inverse of the coarsest-level matrix. AMG methods are characterized by the coarsening procedure applied to setup the sequence of coarse matrices, that is, the corresponding prolongation matrices; they use only information from the original (fine) matrix, with no additional information related to the geometry of the problem. In AMG4PSBLAS two different parallel coarsening procedures for spd matrices are available; both employ disjoint aggregates of fine dofs to form the coarse dofs, and the prolongation matrices are piecewise-constant interpolation matrices (unsmoothed aggregation) or a smoothed variant thereof (smoothed aggregation). Details on the coarsening procedures implemented in AMG4PSBLAS are available in the literature.[7,23] Here we note that in section 6 we refer to the two acronyms:

DSVMB: the smoothed aggregation scheme introduced by Vaněk, Mandel and Brezina in Reference 24, 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.[23]

SMATCH: the smoothed aggregation scheme introduced by D'Ambra and Vassilevski.[7,25,26] 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 so that computational complexity and convergence properties of the final preconditioner can be balanced.

The parallel smoothers available in AMG4PSBLAS can be applied both as one-level preconditioners as well as in an AMG procedure. These include variants of the AS methods described above, weighted versions of the simple Jacobi method, such as the so-called ℓ_1–Jacobi, and a hybrid version of Gauss-Seidel, which acts as the Jacobi method among matrix blocks assigned to different parallel processes and updates the unknowns in a "Gauss-Seidel style" within the block owned by a single process.[7,27] Some of the above methods are well suited for use with the PSBLAS plugin for GPU accelerators in the preconditioner application phase.

6 | NUMERICAL EXPERIMENTS

In this section, we investigate the parallel performances of our Richards equation solver relying on the PSCToolkit software framework using preconditioners discussed in Section 5.2. The section is divided in two parts: in Section 6.1 we analyse *strong scalability* by measuring how the overall computational time decreases with the number of processes for a fixed problem size, then in Section 6.2 we focus on *weak scalability*, that is, we measure how the solution time varies by increasing the number of processes, while keeping fixed the problem size per process, so that the global problem size proportionally increases with the number of processes. Experiments validating the spectral analysis are reported in Supplementary Material Sections E and F.

All the experiments are executed on the CPU cores of the *Marconi-100 supercomputer* (June 21, 2022 TOP500 list[†]), with no usage of hyperthreading. *Marconi*'s nodes are built on an *IBM Power System AC922*, they contain two banks of 16 cores *IBM POWER93* 3.1 GHz processors and are equipped with 256 GB of RAM. The inter-node communication is handled by a Dual-rail *Mellanox EDR Infiniband* network by *IBM* with 220/300 GB/s of nominal and peak frequency. All the code is compiled with the `gnu/8.4.0` suite and linked against the `openmpi/4.0.3` and `openblas/0.3.9` libraries. We use only inner functionalities of the PSBLAS 3.7.0.2 and AMG4PSBLAS 1.0 libraries, with no use of optional third-party libraries.

We discuss the results of the overall solution procedure for solving the discretized Richards equation, by comparing parallel performance and convergence behavior when the AS preconditioner in (13) and the two AMG preconditioners based on the two different coarsening strategies discussed in Section 5.2 are employed for solving the linear systems within the modified Newton procedure. For the one-level *AS* preconditioner we use one layer of mesh points in each direction as overlap among the subdomains Ω_i, each of them assigned to different processes, and apply an Incomplete LU factorization with no fill-in for computing the local subdomain matrix inverses M_i^{-1}. In the case of AMG preconditioners, we apply a symmetric V-cycle with 1 iteration of hybrid backward/forward Gauss-Seidel as pre/post-smoother at the intermediate levels. As a 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. The coarsest-level iterative procedure is stopped when the relative residual is less than 10^{-4} or when a maximum number of 30 iterations is reached. In both the DSVMB and SMATCH procedures, the coarsening is stopped when the size of the coarsest matrix includes no more than 200 dofs per core; in the case of the SMATCH coarsening, a maximum size of aggregates equal to 8 is required. In the following, the two AMG preconditioners are referred as VDSVMB and VSMATCH, respectively. In order to reduce the setup costs of the AMG preconditioners in nonlinear and/or time-dependent simulations, we support a re-use strategy of operators that is often used in our Richards solution approach. Namely, we compute the multilevel hierarchy of coarser matrices only on the first Jacobian of the sequence (at the first time step), then for subsequent Jacobians we just update the smoother on the current approximation of the matrix, while keeping fixed the coarser matrices and transfer operators. Moreover, at each subsequent time step, we reuse the Jacobian (and its related approximation) from the previous one; we leave KINSOL control over the possible need to recompute a Jacobian at each new Newton iteration.

All the preconditioners are applied as right preconditioner to the PSBLAS-based Restarted GMRES with restarting step equal to 10 – GMRES(10). We stop the iterations when the relative residual satisfies $\|J\mathbf{d}_r + \mathbf{\Phi}\| < \eta\|\mathbf{\Phi}\|$ with $\eta = 10^{-7}$ or when a maximum number of iterations equal to 200 is done.

To solve the Richards equation with upstream averages on a parallelepipedal domain Ω of size $[0, L_x] \times [0, L_y] \times [0, L]$, we apply water at height $z = L$ such that the pressure head becomes zero in a square region at the center of the top layer, $\left(\frac{a}{4} \leq x \leq \frac{3a}{4}, \frac{b}{4} \leq y \leq \frac{3b}{4}\right)$, and is fixed to the value $h = h_r$ on all the remaining boundaries, that is

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

where we denote by χ_Ω the characteristic function of the set Ω. The associated 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 $N_t = 10$. This means that the number of time steps N_t is fixed independently from the number of processes n_p, thus the performance analysis will be done on the quantities averaged on the number of time steps relative to the given n_p.

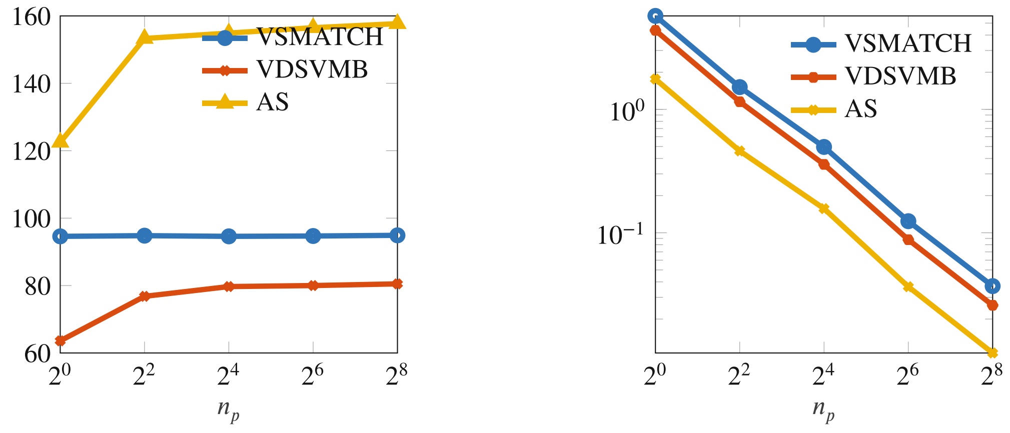
All the timings reported in the figures of the following sections are in seconds.

6.1 | Strong scalability

In this section, we discuss parallel performance results of our solution procedure when we fix the target domain as the parallelepiped $[0, 64] \times [0, 64] \times [0, 1]$, discretized with $N_x = N_y = 800$ mesh points in the x and y directions, and $N_z = 40$ mesh points in the vertical direction, for a total number of 20 millions of dofs, on a number of computational cores from 1 to 256; in particular we used a number of cores $n_p = 4^p$ $p = 0, \ldots, 4$, so that the computational domain is uniformly partitioned in an increasing number of vertical subdomains with square basis, for increasing number of parallel cores.

We start by looking at the average number of linear iterations done for the Newton correction by using the three different preconditioners, and the time needed per each linear iteration in Figure 2. We observe that, as expected, the AMG preconditioners require a smaller number of iterations than the AS method; the latter shows an increase in the number of iterations as the number of cores, and therefore of subdomains, increases. VDSVMB always requires the smallest number of iterations showing the ability of the DSVMB coarsening procedure to setup a good quality matrix hierarchy on the first Jacobian. On the other hand, we observe that when increasing the number of cores, the number of linear iterations also increases, due to the decoupled parallel approach of the coarsening. The VSMATCH algorithm, thanks to its coupled approach, produces a number of iterations that is essentially unaffected by the number of processes, even though it is always larger than VDSVMB.

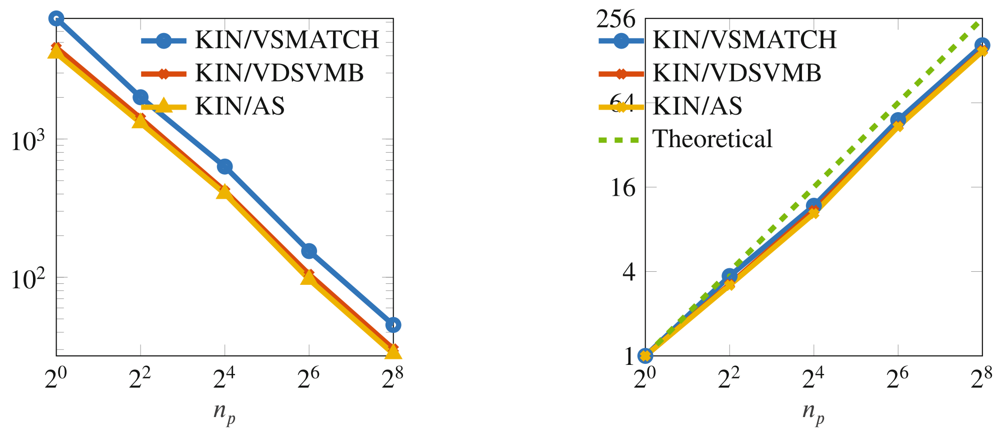
If we look at the time per linear iteration in Figure 2B, we observe, as expected, that the AS preconditioner has the smallest time per iteration. Indeed, for his one-level nature, its application cost is smaller than that of the multigrid methods. It also shows a regular decreasing for increasing number of parallel cores. If we look at the time per iteration of the AMG preconditioners, we observe that VDSVMB and VSMATCH have very similar behavior, and a regular decreasing for larger number of cores. On the other hand, VDSVMB always achieves a smaller time per iteration than VSMATCH, due to its better coarsening ratio. Indeed, it coarsens the original fine matrix in an efficient way, producing coarse matrices that are both smaller and of good quality. This feature makes the VDSVMB preconditioner competitive with respect to the



(a) Average number of linear iteration for the overall simulation

(b) Execution Time per linear iteration $T(s)$

FIGURE 2 Strong scaling. Average number of linear iterations and execution time per linear iteration with different preconditioners.



(a) Total time to solution $T(s)$

(b) Speedup

FIGURE 3 Strong scaling. Total execution time and Speedup for the solution of the whole problem (all time steps), KINSOL with different linear solvers.

AS method. Both preconditioners produce similar global solution times, as shown in Figure 3. We can observe that, in all the cases, the overall simulation has a regular decreasing computational time for increasing number of cores, leading to a global speedup ranging from 150 to 169, depending on the preconditioner, on 256 computational cores. This corresponds to a satisfactory parallel efficiency ranging from 59% to 66%.

For the sake of completeness, in Table 1 we report the total number of computed Jacobians and of the Newton iterations required by the global simulation, when the different preconditioners are employed. We can see that the choice of the preconditioner also affects the KINSOL nonlinear procedure, and that the best behavior always corresponds to the VDSVBM preconditioner.

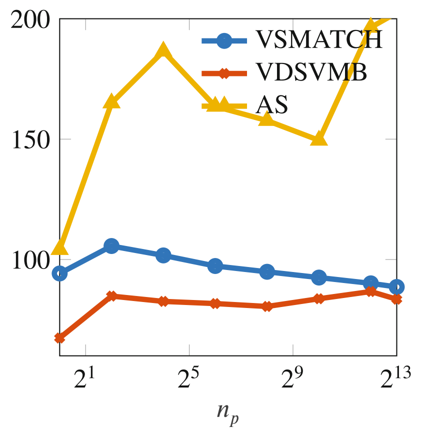
6.2 | Weak scalability

In this section, we look at the *weak scalability* of the overall solution procedure. The ideal goal is to obtain a constant execution time when the number of cores increases, while the computational work per core is kept fixed. Unfortunately, it is, in general, impossible to get an exactly constant execution time. Indeed, we would need perfect *algorithmic* scalability of the preconditioners, that is, the ability to keep the number of linear iterations perfectly constant with an increasing

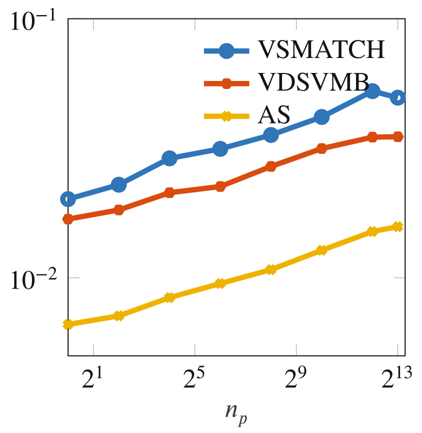
TABLE 1 Strong scaling.

	VDSVBM		VSMATCH		AS	
n_p	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

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



(a) Average number of linear iteration for the overall simulation



(b) Execution Time per linear iteration $T(s)$

FIGURE 4 Weak scaling. Average number of linear iterations and execution time per linear iteration with different preconditioners.

problem size. Moreover, the parallelization efficiency of sparse matrix kernels, which are communication bound, is also affected by the increase in the number of processes and hence in the communication requirements.

To achieve a scaling of the computational work that is meaningful with respect to the physical properties of the underlying problem, we consider a growing domain $\Omega(n_p) = [0, 2^p \times 4.0] \times [0, 2^q \times 4.0] \times [0, 1.0]$ split on $n_p = p \times q$ processes for increasing $p = 0, \dots, 7$, $q = 0, \dots, 6$, and a corresponding mesh with a total of points equal to $N(p \times q) = (2^p N_x, 2^q N_y, N_z)$, where $N_x = N_y = 50$, and $N_z = 40$. In this way, each process has the same amount of computational and communication load, and the overall problem size increases with the total number of processes n_p. We run experiments with up to 8192 CPU cores for solving a problem with a global size of up to about 829 million dofs.

As in the previous section, we first discuss convergence behavior and efficiency of the linear solvers for the Newton corrections, when the different preconditioners are employed. In Figure 4 we show the average number of linear iterations along the overall simulation and the average execution time per each linear iteration. We observe that, as expected, the AS method has the worst algorithmic scalability, showing a general increase in the linear iterations needed to solve problems of increasing size. The average number of iterations has a slow and small decrease from 16 to 1024 cores, while there is a rapid increase from 150 to 203 iterations going from 1024 up to 8192 cores, confirming that convergence properties of the one-level Schwarz-type domain decomposition methods are dependent on the number of involved subdomains. On the other hand, if we look at the AMG preconditioners, where a better coupling among the subdomains is considered, we see that the average number of iterations is smaller and has a very small variation for increasing number of subdomains. In particular, we observe that for the VDSVMB preconditioner the average number of linear iterations has an increase from 67 to 84 iterations going from 1 to 8192 cores. The VSMATCH preconditioner, after an initial small increase, displays a decrease which shows that the coupled aggregation algorithm based on matching is able to produce an effective preconditioner with very good algorithmic scalability properties. This makes VSMATCH promising for exploring extreme scalability in this type of simulation procedure for the Richards model.

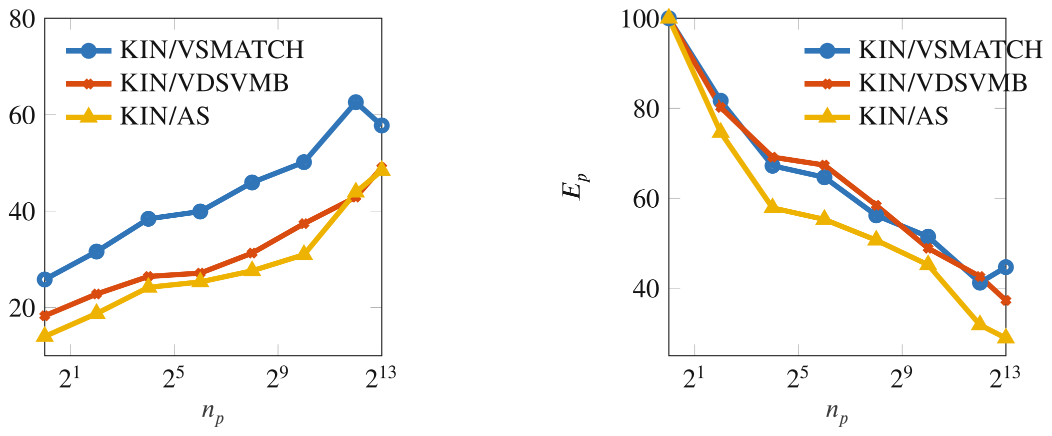
If we look at the average execution time per linear iteration, for all the preconditioners we observe a very small increase, which demonstrates a good implementation scalability for all the sparse computations involved in the linear solver phase.

We now turn to the evaluation of the overall time to solution and *scaled efficiency*[‡], of the global simulation. In Figure 5A, we see that, as for linear solvers, also for the global procedure there is a generally small increase for increasing number of cores and problem size for all the employed preconditioners, except for the VSMATCH method going from 4096 to 8192 cores, where a better convergence behavior of the linear solver also produces a small reduction in nonlinear iterations (cf. Table F3 in the Supplementary Material). The best time to solution is in general obtained by using the AS preconditioner till to 1024 cores, due to the smallest computational cost of this preconditioner. On the other hand, its worst algorithmic scalability (i.e., the large increase in the number of linear iterations) results in larger global execution times with respect to the most effective VDSVMB preconditioner when the number of cores increases.

If we look at the scaled efficiency shown in percentage scale in Figure 5, the AMG preconditioners show similar behavior, while VSMATCH confirms its better efficiency when the largest number of cores is used.

To give a complete picture of the performance of the various parts of the solution procedure we also report the percentage of the execution time spent in the different computational kernels. The barplots in Figure 6 deliver several useful information. First of all, as expected, the two main parts of the efforts are represented by the time spent in actually solving the linear systems (6) (LinSol) and evaluating the nonlinear function $\mathbf{\Phi}$ encoding the discretization (Feval), on which our implementation efforts were more focused. In particular, in the Feval kernel we exploited many of the support functionalities of PSCToolkit. To increase the efficiency, at the expense of a marginal increase in the use of memory, we experimented with memorizing the *local-to-global* map on the MPI tasks, thus avoiding the overhead of explicit calls to the functions that permit the user to explore the index space. This is made possible by the fact that we are using a communicator for data distribution which is fixed through the time steps and iterates of Newton's method; see the discussion in Section 5.1. Thanks to the flexibility of the underlying software framework, we were able to optimize the procedure for building the vectors and matrices at each nonlinear time step, making good reuse of local data.

We have a small and almost equal time that is spent in building the distributed Jacobian (Jacobian) and auxiliary matrices (Auxiliary). The rebuild of the local portions of the matrices is well optimized and is very close to the absolute minimum that is necessary to compute the coefficient values. A reasonably small amount of time is spent in building and updating the preconditioners (Setup). In particular, it is possible to reuse and apply the aggregation hierarchy, so that we only need to recompute the smoothers. The other (almost invisible) bar is represented by the halo exchanges: this is the data exchange among the parallel processes that happens before each build of the Jacobian, build of auxiliary matrices, and function evaluation. It is a communication that is necessary for having an agreement on all the quantities needed by the processes to perform their computations, namely the exchange of the boundary data; this operation is persistent,



(a) Total time to solution $T(s)$ **(b)** Scaled Efficiency

FIGURE 5 Weak scaling: Total execution time and Scaled efficiency shown in percentage for the solution of the whole problem (all time steps), KINSOL with different linear solvers.

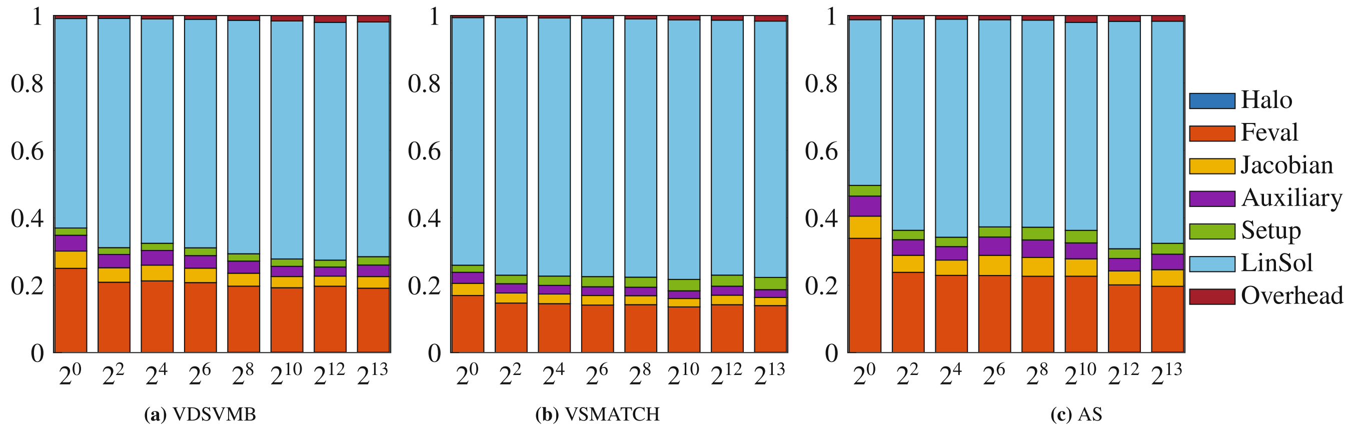


FIGURE 6 Percentage of the overall time spent in the different phases for the overall solution process. We sum all the partial times and normalize it against the total to account for the different computational kernels.

in the sense that the pattern of the data exchange is determined by the discretization mesh structure. The last overhead is all the remaining computations that are made inside the Newton method, that is, vector updates and vector norm computations, namely the Newton direction updates, convergence checks, and line-searches, and are in charge to the KINSOL library.

7 | CONCLUSIONS AND PERSPECTIVES

In this paper, we focused on two main objectives: to prove some spectral properties of the sequence of Jacobian matrices generated by discretizing the Richards equation in mixed form for simulation of unsaturated subsurface flows by a Newton-type method, and to prove the efficiency, flexibility, and robustness of a software framework for parallel sparse matrix computations.

The theoretical results we obtained are consistent with expectations and justify some preconditioner choices already used in the literature for solving Richards equations. We used several functionalities of the PSCToolkit for iterative solution of sparse linear systems, some of the support routines for re-using preconditioners in solving sequences of linear systems, and for an efficient setup and update of data structures needed for Jacobian and right-hand side computations in Newton iterations. The performances of our strategy on one of the most powerful supercomputers currently available are quite promising in view of exploring extreme scalability and confirming the benefits of using multigrid preconditioners when the number of processing cores largely increases.

Our plans for future work include the extension of the PSCToolkit interface to KINSOL, in order to use the ability of the PSCToolkit linear solvers in exploiting GPU architectures, and the integration of the software stack into the Parflow code[28] for realistic simulations in hydrological applications.

ACKNOWLEDGMENTS

We thank Prof. Stefan Kollet from the Jülich Supercomputing Center for the information regarding the implementation of the Richards equation in PARFLOW and the insightful discussions regarding its discretization. We acknowledge PRACE (Partnership for Advanced Computing in Europe) for awarding us access to the high-performance computing system Marconi100 at CINECA (Consorzio Interuniversitario per il Calcolo Automatico dell'Italia Nord-orientale) under the grants Pra22_EoCoE, Pra23_EoCoE and Pra24_EoCoE. We also thank the two anonymous reviewers for their helpful comments which enabled us to improve the presentation and clarity of the manuscript.

FUNDING INFORMATION

This work has been partially supported by EU under the Horizon 2020 Project "Energy oriented Centre of Excellence: toward exascale for energy (EoCoE-II), Project ID: 824158. All the authors are member of the INdAM Research Group GNCS.

CONFLICT OF INTEREST STATEMENT

The authors declare no potential conflict of interests.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in PSCToolkit at https://github.com/psctoolkit/psctoolkit.

ENDNOTES

*The libraries can be obtained from https://psctoolkit.github.io/.

†See the relevant list on the website https://www.top500.org/.

‡Scaled efficiency is defines as $E_p = \frac{T_1(N)}{T_p(N \cdot n_p)}$, where $T_1(N)$ is the time employed to solve a problem of size N on 1 processing unit, and $T_p(N \cdot n_p)$ is the time employed for solving a problem of size $N \times n_p$ on n_p processing units.

ORCID

Daniele Bertaccini https://orcid.org/0000-0002-3662-278X
Pasqua D'Ambra https://orcid.org/0000-0003-2047-4986
Fabio Durastante https://orcid.org/0000-0002-1412-8289
Salvatore Filippone https://orcid.org/0000-0002-5859-7538

REFERENCES

1. Farthing MW, Ogden FL. Numerical solution of Richards' equation: a review of advances and challenges. Soil Sci Soc Am J. 2017;81(6):1257–69. Available from: https://acsess.onlinelibrary.wiley.com/doi/abs/10.2136/sssaj2017.02.0058
2. van Genuchten MT. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Sci Soc Am J. 1980;44(5):892–8. Available from: https://acsess.onlinelibrary.wiley.com/doi/abs/10.2136/sssaj1980.03615995004400050002x
3. Celia MA, Bouloutas ET, Zarba RL. A general mass-conservative numerical solution for the unsaturated flow equation. Water Resour Res. 1990;26(7):1483–96. Available from: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/WR026i007p01483
4. Miller CT, Williams GA, Kelley CT, Tocci MD. Robust solution of Richards' equation for nonuniform porous media. Water Resour Res. 1998;34(10):2599–610. Available from: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/98WR01673
5. Jones JE, Woodward CS. Newton-Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. Adv Water Resour. 2001;24(7):763–74. Available from: http://www.sciencedirect.com/science/article/pii/S0309170800000750
6. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, et al. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Trans Math Soft. 2005;31(3):363–96.
7. D'Ambra P, Durastante F, Filippone S. AMG preconditioners for linear solvers at extreme scale. SIAM J Sci Comp. 2021;43(5):S679–S703.
8. Haverkamp R, Vauclin M, Touma J, Wierenga PJ, Vachaud G. A comparison of numerical simulation models for one-dimensional infiltration. Soil Sci Soc Am J. 1977;41(2):285–94. Available from: https://acsess.onlinelibrary.wiley.com/doi/abs/10.2136/sssaj1977.03615995004100020024x
9. Brooks RH, Corey AT. Hydraulic properties of porous media. Vol 3. Hydrology papers, Fort Collins, Colorado: Colorado State University; 1964.
10. Szymkiewicz A. Approximation of internodal conductivities in numerical simulation of one-dimensional infiltration, drainage, and capillary rise in unsaturated soils. Water Resour Res. 2009;45(10):1–16. Available from: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2008WR007654
11. Baker DL. General Validity of Conductivity Means in Unsaturated Flow Models. J Hydrol Eng. 2006;11(6):526–38.
12. Haverkamp R, Vauclin M. A note on estimating finite difference interblock hydraulic conductivity values for transient unsaturated flow problems. Water Resour Res. 1979;15(1):181–7. Available from: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/WR015i001p00181
13. Oldenburg CM, Pruess K. On numerical modeling of capillary barriers. Water Resour Res. 1993;29(4):1045–56. Available from: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/92WR02875
14. Srivastava R, Guzman-Guzman A. Analysis of hydraulic conductivity averaging schemes for one-dimensional steady-state unsaturated flow. Ground Water 1995. 2020;33:946+.
15. Garoni C, Serra-Capizzano S. Generalized locally Toeplitz sequences: theory and applications: Volume I. Cham: Springer International Publishing; 2017 Available from: https://doi.org/10.1007/978-3-319-53679-8
16. Garoni C, Serra-Capizzano S. Generalized locally Toeplitz sequences: theory and applications. Vol II. Cham: Springer; 2018 Available from: https://doi.org/10.1007/978-3-030-02233-4
17. Kollet SJ, Maxwell RM. Integrated surface-groundwater flow modeling: a free-surface overland flow boundary condition in a parallel groundwater flow model. Adv Water Resour. 2005;29(7):945–958. Available from: https://www.osti.gov/biblio/887270
18. Maxwell RM. A terrain-following grid transform and preconditioner for parallel, large-scale, integrated hydrologic modeling. Adv Water Resour. 2013;53:109–17.

19. Vassilevski PS. Multilevel block factorization preconditioners: matrix-based analysis and algorithms for solving finite element equations. New York: Springer; 2008.
20. Filippone S, Colajanni M. PSBLAS: a library for parallel linear algebra computations on sparse matrices. ACM TOMS. 2000;26(4):527–50.
21. Filippone S, Buttari A. Object-oriented techniques for sparse matrix computations in Fortran 2003. ACM TOMS. 2012;38(4):23:1–23:20.
22. Buttari A, D'Ambra P, di Serafino D, Filippone S. 2LEV-D2P4: a package of high-performance preconditioners for scientific and engineering applications. AAECC. 2007;18(3):223–39.
23. D'Ambra P, di Serafino D, Filippone S. MLD2P4: a package of parallel algebraic multilevel domain decomposition preconditioners in Fortran 95. ACM Trans Math Softw. 2010;37(3):30.
24. Vaněk P, Mandel J, Brezina M. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. Computing. 1996;56(3):179–96. International GAMM-Workshop on Multi-level Methods (Meisdorf, 1994).
25. D'Ambra P, Vassilevski PS. Adaptive AMG with coarsening based on compatible weighted matching. Comput Vis Sci. 2013;16(2):59–76.
26. D'Ambra P, Filippone S, Vassilevski PS. BootCMatch: a software package for bootstrap AMG based on graph weighted matching. ACM Trans Math Softw. 2018; 44(4):1–25. Available from: https://doi.org/10.1145/3190647
27. D'Ambra P, Filippone S. A parallel generalized relaxation method for high-performance image segmentation on GPUs. J Comput Appl Math. 2016;293:35–44.
28. Maxwell RM, Condon LE, Smith SG, Woodward CS, Falgout RD, Ferguson IM, et al. Integrated GroundWater modeling center report. ParFlow User's Manual Zenodo. Princeton, New Jersey: Integrated Groundwater Modeling Center; 2022. p. 171 Available from: https://github.com/parflow/parflow/blob/master/parflow-manual.pdf

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Bertaccini D, D'Ambra P, Durastante F, Filippone S. Why diffusion-based preconditioning of Richards equation works: Spectral analysis and computational experiments at very large scale. Numer Linear Algebra Appl. 2023;e2523. https://doi.org/10.1002/nla.2523