Bergamaschi, L.; Bru García, R.; Martinez Calomardo, A.; Mas Marí, J.; Putti, M. (2013). Low-rank update of preconditioners for the nonlinear Richard's equation. Mathematical and Computer Modelling. 57(7):1933-1941. doi:10.1016/j.mcm.2012.01.013.

# Low-rank Update of Preconditioners for the nonlinear Richards Equation.

L. Bergamaschi[(1)], R. Bru[(2)], A. Martínez[(1)], J. Mas[(2)], and M. Putti[(1)]

[(1)] *Department of Mathematical Methods and Models for Scientific Applications University of Padua, via Trieste 63, 35121 Padova, Italy*
e-mail {*berga,acalomar,putti*}*@dmsa.unipd.it,*
[(2)] *Instituto de Matemática Multidisciplinar, Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, Spain,* e-mail {*rbru,jmasm*}*@mat.upv.es.*
*Work supported by the Spanish DGI grant MTM2010-18674*

**Abstract**

Preconditioners for the Conjugate Gradient method are studied to solve the Newton system with symmetric positive definite (SPD) Jacobian. Following the theoretical work in [4] we start from a given approximation of the inverse of the initial Jacobian, and we construct a sequence of preconditioners by means of a low rank update, for the linearized systems arising in the Picard-Newton solution of the nonlinear discretized Richards equation. Numerical results onto a very large and realistic test case show that the proposed approach is more efficient, in terms of iteration number and CPU time, as compared to computing the preconditioner of choice at every nonlinear iteration.

*Keywords:* Quasi-Newton method, Krylov iterations, updating preconditioners, inexact Newton method

## 1. Introduction

The governing equation for flow in partially saturated porous media, Richards equation, contains nonlinearities arising from pressure head dependencies in soil moisture and hydraulic conductivity. For stability reasons an implicit time discretization, requiring evaluation of the nonlinear coefficients at the current time level, is normally used to solve the equation numerically. To linearize the resulting discrete system of equations, Newton or Picard iteration is commonly used, with the Picard scheme being the more popular of the two [17, 14, 8]. The Picard method, also known as successive approximation or "simple" iteration, is computationally inexpensive on a per iteration basis, and preserves symmetry of the discrete system of equations.

In this paper we are mainly concerned with the efficient preconditioning of the linear system arising in the Picard iteration for the solution of a general system of nonlinear equations: $\boldsymbol{F}(\boldsymbol{x}) = 0$, which is usually written as

$$\left\{ \begin{array}{lcl} J(\boldsymbol{x}_k)\boldsymbol{s}_k & = & -\boldsymbol{F}(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1} & = & \boldsymbol{x}_k + \boldsymbol{s}_k \end{array} \right. . \tag{1}$$

We employ the Preconditioned Conjugate Gradient (PCG) method for the solution of the linear system, so that two nested iterative procedures need to be implemented, the outer iteration formed by the Newton steps and the inner iterations within the PCG method.

Since in each Newton step a new system has to be solved, we are dealing with the construction of a sequence of preconditioners $P_k \approx J_k^{-1}$ which are "optimal" in the sense that they would minimize the constant $C$ of:

$$\|I - P_k J(\boldsymbol{x}_k)\| \leq C. \tag{2}$$

Following the previous work in [4, 5, 6], our aim is to solve the Picard SPD system (1) with the PCG method, starting with an initial preconditioner, $P_0$, computed from the initial Jacobian, and to update this preconditioner using low-rank matrices. We have selected $P_0$ to be IC(0) or AINV [3, 2] as popular representants of factorized direct and inverse based preconditioners. A sequence of SPD preconditioners $P_k$ can thus be defined by imposing the secant condition, as used in the implementation of quasi-Newton methods. We choose to work with the BFGS update as described for instance in [12], and recall the theoretical properties of the preconditioner and numerical behavior of the resulting scheme.

We stress that our theoretical developments will make our procedure free from the initial preconditioner choice. If a better performing initial $P_0$ than IC(0) (or AINV) is devised, then the rank one update formula will improve it, may be even better than it is shown in this paper.

We show results on a very large realistic test case where the proposed sequence of preconditioners provides an improvement of the iteration number and CPU time as compared to simply recomputing the preconditioner of choice.

## 2. The Mathematical Model

Richards equation is commonly used in modeling the flow of water in the unsaturated zone, and is obtained by combining Darcy's law with the continuity equation. For practical purposes, it is convenient to use a pressure head formulation so that the simultaneous simulation of both fully saturated ($S_w = 1$) and unsaturated ($S_w < 1$) conditions is possible [11].

The water flux $\vec{q}$ is governed by the extended form of Darcy's law [16, 1]:

$$\vec{q} = -K_s K_r(\psi) \vec{\nabla} (\psi + z) \tag{3}$$

where $K_s$ is the saturated hydraulic conductivity tensor, $K_r(\psi)$ is the relative hydraulic conductivity, $\vec{\nabla}$ is the gradient operator, and $z$ is the vertical coordinate directed upward. The mass conservation equation governing flow in variably saturated porous media is called Richards equation, and can be written as:

$$\eta(S_w) \frac{\partial \psi}{\partial t} + \vec{\nabla} \cdot \vec{q} = f \tag{4}$$

where $\eta(S_w) = S_w S_s + \phi \partial S_w / \partial \psi$ is the general storage term, $S_w$ is the water saturation, $S_s$ is the elastic storage coefficient, $\phi$ is the porosity of the porous medium, $t$ is time, and $f$ represents the source or sink term. Equation (3) is nonlinear because of the dependency of the relative conductivity on pressure head, while Equation (4) assumes its nonlinearity in the general storage term, a function of pressure head through the $S_w - \psi$ relationship. The nonlinear functions $K_r(\psi)$ and $S_w(\psi)$ in (3) and (4), respectively, can be modeled using different characteristic curves that are generally determined experimentally for different soils [18]. We use the van Genuchten expression ([14]) given by:

$$\begin{array}{lcll} S_w(\psi) & = & S_{wr} + (\phi - S_{wr}) \left[1 + \frac{\psi}{\psi_s}\right]^{-m} & \psi < 0 \\ S_w(\psi) & = & 1 & \psi \geq 0 \end{array} \tag{5}$$

$$\begin{aligned} K_r(\psi) &= (1+\tfrac{\psi}{\psi_s})^{-5m/2}\left[(1+\tfrac{\psi}{\psi_s})^m - \left(\tfrac{\psi}{\psi_s}\right)^m\right]^2 \qquad \psi < 0 \\ K_r(\psi) &= 1 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \psi \geq 0 \end{aligned} \tag{6}$$

where $S_{wr}$ is the residual saturation, $\psi_s$ is the air-entry pressure value, and $m$ is a fitting parameter.

Initial and boundary conditions have to be specified to complete the mathematical model. Boundary conditions may be of Dirichlet (prescribed pressure head) or Neuman (prescribed flux) type. For practical applications, it is necessary to have boundary conditions that vary in time. In the simulation of rainfall or evaporation, the so called "atmospheric boundary conditions", alternation of infiltration and exfiltration may cause the boundary conditions to switch from Dirichlet to Neuman type, depending on the prevailing flux and pressure head values at the surface [15].

## 3. Numerical Procedures

### *3.1. Governing Equation and Finite Element Models*

The partial differential equation describing fluid flow in partially saturated porous media, Richards equation, is obtained by combining Darcy's law with the continuity equation [16]. Expressing this equation with pressure head $\psi$ as the dependent variable, $t$ as time, and $z$ as the vertical coordinate (positive upward) yields

$$\eta(\psi)\frac{\partial \psi}{\partial t} = \nabla \cdot (K_s K_r(\psi)\, \nabla(\psi + z)) \tag{7}$$

where $\eta(\psi)$ is the general storage term or overall storage coefficient and the hydraulic conductivity tensor is expressed as a product of the conductivity at saturation, $K_s$ and the relative conductivity, $K_r(\psi)$. Equation (7) is highly nonlinear due to pressure head dependencies in the storage and conductivity terms.

To solve equation (7) numerically, a finite element Galerkin discretization in space with linear basis functions is used. Triangular elements are used in the two-dimensional code, and tetrahedral elements in three dimensions. With tetrahedra the nonlinear coefficients in the system integrals are evaluated at the element centroids. A $\lambda$-weighted finite difference scheme is used for time discretization ($\lambda = 0.5$, Crank-Nicolson; $\lambda = 1$, backward Euler). Discretization yields the system of nonlinear equations

$$\boldsymbol{F}(\boldsymbol{\Psi}^{n+1}) \equiv A(\boldsymbol{\Psi}^{n+\lambda})\boldsymbol{\Psi}^{n+\lambda} + P(\boldsymbol{\Psi}^{n+\lambda})\frac{\boldsymbol{\Psi}^{n+1} - \boldsymbol{\Psi}^{k}}{\Delta t^{n+1}} - \boldsymbol{b}(\boldsymbol{\Psi}^{n+\lambda}) - \boldsymbol{q}(t^{n+\lambda}) = \boldsymbol{0} \tag{8}$$

where $\boldsymbol{\Psi}^{n+\lambda} = \lambda\boldsymbol{\Psi}^{n+1} + (1-\lambda)\boldsymbol{\Psi}^{k}$, $\boldsymbol{\Psi}$ is the vector of nodal pressure heads, superscript $n$ denotes time step, $A$ is the stiffness matrix, $P$ is the storage or mass matrix, $\boldsymbol{b}$ contains the gravitational gradient component of equation (7), and $\boldsymbol{q}$ contains the specified Darcy flux boundary conditions.

The numerical models have the option of using either distributed or lumped mass matrices. The models can handle a variety of boundary conditions, including atmospheric inputs, seepage faces, and source/sink terms such as pumping wells.

### *3.2. Linearization Techniques: Newton and Picard Iteration*

Applied to equation (8), the Newton scheme can be written as

$$J(\boldsymbol{\Psi}_k^{n+1})\boldsymbol{s}_k = -\boldsymbol{F}(\boldsymbol{\Psi}_k^{n+1}) \tag{9}$$

where $\boldsymbol{s}_k \equiv \boldsymbol{\Psi}_{k+1}^{n+1} - \boldsymbol{\Psi}_k^{n+1}$, $k$ is the iteration index, and

$$\begin{aligned} J_{ij} &= \lambda A_{ij} + \frac{1}{\Delta t^{n+1}} P_{ij} + \sum_s \frac{\partial A_{is}}{\partial {\psi_j}^{n+1}} {\psi_s}^{n+\lambda} \\ &\quad + \frac{1}{\Delta t^{n+1}} \sum_s \frac{\partial P_{is}}{\partial {\psi_j}^{n+1}} ({\psi_s}^{n+1} - \psi_s^n) + \frac{\partial b_i}{\partial {\psi_j}^{n+1}} \end{aligned} \tag{10}$$

is the $ij$-th component of the Jacobian matrix $J(\boldsymbol{\Psi}^{n+1})$.

The Picard scheme may be written as

$$\left[ \lambda A(\boldsymbol{\Psi}_k^{n+\lambda}) + \frac{1}{\Delta t^{n+1}} P(\boldsymbol{\Psi}_k^{n+\lambda}) \right] \boldsymbol{s}_k = -\boldsymbol{f}(\boldsymbol{\Psi}_k^{n+1}) \tag{11}$$

Comparing (9) and (11), it is apparent that the Picard scheme can be viewed as an approximate Newton method. It is well known that under suitable conditions the Newton scheme is quadratically convergent, while Picard converges only linearly. Another important difference between the two schemes is that Newton linearization generates a nonsymmetric system matrix, whereas Picard preserves the symmetry of the original discretization. A final observation to make is that calculation of the three derivative terms in the Jacobian makes the Newton scheme more costly and algebraically complex than Picard.

Time step sizes during a transient simulation are dynamically adjusted according to the convergence behavior of the nonlinear iteration scheme.

## 4. BFGS-update of preconditioners and convergence analysis

The idea is to start with a preconditioner $P_0 \approx J_0^{-1}$. Let us define $\boldsymbol{y}_k = \boldsymbol{F}(\boldsymbol{x}_{k+1}) - \boldsymbol{F}(\boldsymbol{x}_k)$ and recall that $\boldsymbol{s}_k$ is the solution of the $k$th Newton system. Following [4] we can develop a similar recurrence formula for the preconditioner as

$$P_{k+1} = \left( I - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\boldsymbol{s}_k^T \boldsymbol{y}_k} \right) P_k \left( I - \frac{\boldsymbol{y}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{y}_k} \right) + \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{y}_k}. \tag{12}$$

If the Jacobian matrices are SPD and so is $P_0$, then $P_k$ is also SPD under the condition $\boldsymbol{s}_k^T \boldsymbol{y}_k > 0$ (see Lemma 4.1.1 in [13]).

It can also be easily proved that the sequence of matrices just defined satisfies the secant condition: $P_{k+1} \boldsymbol{y}_k = \boldsymbol{s}_k$.

We will prove that $\|I - P_k J(\boldsymbol{x}_k)\|$ can be tuned to any fixed accuracy by suitable choices of the initial guess $\boldsymbol{x}_0$ and the initial preconditioner $P_0$. Note that this makes our preconditioner almost ideal in the sense of (2). Let us denote with $\Omega$ an open subset of $\mathcal{R}^n$, we will make the following *standard assumptions* on $\boldsymbol{F}$ which we will assume to hold throughout this section.

1. Equation $\boldsymbol{F}(\boldsymbol{x}) = 0$ has a solution $\boldsymbol{x}^*$.
2. $J(\boldsymbol{x}) : \Omega \to \mathcal{R}^{n \times n}$ is Lipschitz continuous with Lipschitz constant $\gamma$.
3. $J(\boldsymbol{x}^*)$ is nonsingular.

Moreover, we assume in the sequel that our preconditioner accelerates the Newton systems in the framework of the Inexact Newton methods [10]. We therefore stop the linear iteration as soon as the following test is satisfied

$$\|J(\boldsymbol{x}_k)\boldsymbol{s}_k + \boldsymbol{F}(\boldsymbol{x}_k)\| \leq \eta_k \|\boldsymbol{F}(\boldsymbol{x}_k)\|. \tag{13}$$

If we choose the sequence $\{\eta_k\}$ such that for every iteration index $k$, $\eta_k = O(\|\boldsymbol{F}(\boldsymbol{x}_k)\|)$, then the following result holds:

**Proposition 4.1.** *Define $\boldsymbol{e}_k = \boldsymbol{x}^* - \boldsymbol{x}_k$. There exist $\delta > 0$ and $r < 1$ such that if $\|\boldsymbol{e}_0\| < \delta$ then $\|\boldsymbol{e}_{k+1}\| \leq r\|\boldsymbol{e}_k\|$ for every $k$.*

We define the error vectors $\boldsymbol{e}_{k+1} = \boldsymbol{x}^* - \boldsymbol{x}_{k+1}$, $\boldsymbol{e}_k = \boldsymbol{x}^* - \boldsymbol{x}_k$ and the error matrices $E_{k+1} = P_{k+1} - J(\boldsymbol{x}^*) = P_{k+1}$, $E_k = P_k - J(\boldsymbol{x}^*)^{-1}$. We start with a Lemma whose proof follows from Theorem 1.2.1 in [12], also known as Banach lemma.

**Lemma 4.2.** *Let $\|\boldsymbol{e}_k\| \leq \delta_0 < \dfrac{1}{\gamma}$. Then $\|J_k^{-1} - J(\boldsymbol{x}^*)^{-1}\| \leq \dfrac{\gamma\delta_0}{1-\gamma\delta_0}$.*

We now state the following important Lemma (whose proof can be found in [4]) which bounds the norm of $E_{k+1}$ in terms of the norm of $E_k$ and $\boldsymbol{e}_k$. In addition, it states that the sequence of preconditioners constructed with the equation (12) is SPD provided that $P_0$ is SPD.

**Lemma 4.3.** *If $P_k$ is SPD and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - J_k^{-1}\boldsymbol{F}_k$ then there is $\delta_0$ such that if $0 < \|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq \delta_0$ and $\|E_k\| \leq \delta_0$ then $\boldsymbol{y}^T\boldsymbol{s} > 0$ showing that $P_{k+1}$ is SPD. Moreover, for some $K > 0$,*

$$\|E_{k+1}\| \leq \|E_k\| + K\|\boldsymbol{e}_k\|.$$

The next result will prove that we can make $\|I - P_k J(\boldsymbol{x}_k)\|$ as small as required. To this end we need the following two Lemmas which bound the difference between $P_k$ and $J(\boldsymbol{x}_k)^{-1}$.

**Lemma 4.4.** *Let $\|\boldsymbol{e}_k\| \leq \delta$. Then setting $E'_{k+1} = P_{k+1} - J_{k+1}^{-1}$, and $E'_k = P_k - J_k^{-1}$ we have, for some $K_1 > 0$, $\|E'_{k+1}\| \leq \|E'_k\| + K_1\|\boldsymbol{e}_k\|$.*

**Lemma 4.5.** *For fixed $\varepsilon > 0$, there are $\delta_0$, $\delta_B$ such that if $\|\boldsymbol{e}_0\| < \delta_0$ and $\|E'_0\| < \delta_B$ then $\|E'_k\| < \varepsilon$, $\forall k > 0$.*

We have just proved that if the initial Newton point $\boldsymbol{x}_0$ is sufficiently close to the solution, and $P_0$ sufficiently close to $J_0^{-1}$, then $E'_k$ can be made as small as desired.

The following theorem establishes that $\|I - P_k J_k\|$ can be made arbitrarily small provided that $\|\boldsymbol{e}_0\|$ and $\|E'_0\|$ are sufficiently small.

**Theorem 4.6.** *For fixed $\varepsilon_1 > 0$, there are $\delta_0$ and $\delta_B$ such that if $\|\boldsymbol{e}_0\| < \delta_0$ and $\|E'_0\| < \delta_B$ then $\|I - P_k J_k\| < \varepsilon_1$.*

*Proof.* Using Lemma 4.5 and the Lipschitz continuity of $J(\boldsymbol{x})$ we have:

$$\|I - P_k J_k\| = \|\left(J_k^{-1} - P_k\right) J_k\| \leq \|J_k\| \cdot \|E'_k\| \leq (1 + \gamma\|\boldsymbol{e}_{k-1}\|)\,\varepsilon \leq (1 + \gamma\delta_0)\varepsilon. \tag{14}$$

If $\delta_0 < \dfrac{1}{2\gamma}$ we can choose $\varepsilon = \dfrac{2}{3}\varepsilon_1$, and $\delta_0, \delta_B$ as in Lemma 4.5 so that the thesis holds. □

## 5. Implementation of the BFGS preconditioner update

At a certain nonlinear iteration level, $k$, and given a vector $\boldsymbol{z}_k^{(l)}$, we want to compute $\boldsymbol{c} = P_k\boldsymbol{z}_k^{(l)}$, where superscript $l$ indicates the linear iteration index. Let us suppose we compute an initial preconditioner $P_0$. Then, at the initial nonlinear iteration $k = 0$, we simply have $\boldsymbol{c} = P_0\boldsymbol{z}_0^{(l)}$.

For $k \geq 0$, $P_{k+1}$ is given inductively by (12). Application of preconditioner $P_k$ to the vector $\boldsymbol{z}_k^{(l)}$ can be performed at the price of $2k$ dot products and $2k$ daxpys as depicted in Figure 1 (for $k = 1$) and in Figure 2 (for a generic $k$), where $\alpha_k = \boldsymbol{s}_k^T\boldsymbol{y}_k$. Note that the updating procedure described above, being based on scalar products and daxpy operations, is well suited to parallelization.

$$a = \frac{\boldsymbol{s}_0^T \boldsymbol{z}_1^{(l)}}{\alpha_0}$$
$$\boldsymbol{w} = \boldsymbol{z}_1^{(l)} - a\boldsymbol{y}_0$$
$$\boldsymbol{w} := P_0 \boldsymbol{w}$$
$$b = \frac{\boldsymbol{y}_0^T \boldsymbol{w}}{\alpha_0}$$
$$\boldsymbol{c} = (a-b)\boldsymbol{s}_0 + \boldsymbol{w}$$

Figure 1: Computation of $\boldsymbol{c} = P_1 \boldsymbol{z}_1^{(l)}$ for the BFGS preconditioner.

$\boldsymbol{w} = \boldsymbol{z}_k^{(l)}$
FOR $r := k-1$ TO $0$
  $a_r = \boldsymbol{s}_r^T \boldsymbol{w}/\alpha_r$
  $\boldsymbol{w} := \boldsymbol{w} - a_r \boldsymbol{y}_r$
END FOR
$\boldsymbol{c} = P_0 \boldsymbol{w}$
FOR $r := 0$ TO $k-1$
  $b = \boldsymbol{y}_r^T \boldsymbol{c}/\alpha_r$
  $\boldsymbol{c} := \boldsymbol{c} + (a_r - b)\boldsymbol{s}_r$
END FOR

Figure 2: Computation of $\boldsymbol{c} = P_k \boldsymbol{z}_k^{(l)}$ for the BFGS preconditioner.

*5.1. Restart*

This kind of algorithm suffers from two main drawbacks, namely the increasing cost of memory for saving $\boldsymbol{y}_k$ and $\boldsymbol{s}_k$ and the increasing CPU time to apply the preconditioner.

Note that these drawbacks are common to many iterative schemes, such as for example sparse (Limited Memory) Broyden implementations. We define $k_{\max}$ the maximum number of rank two corrections we allow. When the nonlinear iteration counter $k$ is larger than $k_{\max}$, the vectors $\boldsymbol{s}_i, \boldsymbol{y}_i,\ i = k \bmod k_{\max}$ are replaced with the last computed $\boldsymbol{s}_k, \boldsymbol{y}_k$ and a new preconditioner $P_0$ is computed. Vectors $\{\boldsymbol{s}_i, \boldsymbol{y}_i\}$ are stored in a matrix $V$ with $n$ rows and $2 \times k_{\max}$ columns.

RESTARTED NEWTON-BFGS (RNBFGS) ALGORITHM
**Input:** $\boldsymbol{x}_0, \boldsymbol{F}, k_{\max}, nlmax$, tol

- $k_{aux} := 0,\ k := 0$
- WHILE $\|\boldsymbol{F}(\boldsymbol{x}_k)\| > \text{tol}\ \|\boldsymbol{F}(\boldsymbol{x}_0)\|$ AND $k < nlmax$ DO
  1. IF $k_{aux} = 0$ THEN Compute $P_0$ approximating $J_0^{-1}$
     ELSE Update $P_k$ from $P_{k-1}$ using the columns of $V$.
  2. Solve $J(\boldsymbol{x}_k)\boldsymbol{s}_k = -\boldsymbol{F}(\boldsymbol{x}_k)$ by a Krylov method with preconditioner $P_k$ and tolerance $\eta_k$.
  3. $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \boldsymbol{s}_k$
  4. $V(*, 2k_{aux}+1) := \boldsymbol{s}_k,\ V(*, 2k_{aux}+2) := \boldsymbol{y}_k,$
  5. $k := k+1\ \ k_{aux} = k \bmod k_{\max}$
- END WHILE

**Output**: $nlit := k, \boldsymbol{x}_k, \|\boldsymbol{F}(\boldsymbol{x}_k)\|$

## 6. Numerical Results

Here we give the numerical performance of our sequence of preconditioners in solving a realistic test case resulting from e 3d FE discretization of the Richards equation. The simulations

were run on a Intel I7 Core 2 QUAD workstation running at 2.4 GHz with 16 Gb RAM. The CPU times are measured in seconds. In the solution of the systems (1) we employed the PCG iterative method and stop the iteration whenever the exit test (13) with constant $\eta_k = 10^{-4}$ is fulfilled. The nonlinear iteration is stopped whenever $\|\boldsymbol{F}(\boldsymbol{x}_k)\| \leq 10^{-8}\|\boldsymbol{F}(\boldsymbol{x}_0)\|$ (i. e. tol = $10^{-8}$). In the subsequent tables we report the total number of nonlinear and linear iterations and CPU time. When $k_{\max} = 0$ we assume that the initial preconditioner is computed, and not updated, at each nonlinear iteration.

*6.1. Test case description*

The test case considers water infiltration from a surface drain into an initially dry soil connected on one side to a channel. The domain is formed by a flat 50 m×50 m terrain overlying a 2 m deep sandy soil. The drain is simulated using a constant flux Neumann boundary condition, while the channel is kept at a constant pressure head. No flow boundary conditions are imposed in all the other parts of the frontier. The sandy soil is characterized by a uniform and isotropic saturated conductivity value $K_s = 10^{-3}$ m/s and a porosity $\phi = 0.3$. The van Genuchten retention curves (5) and (6) are used in the unsaturated zone, with values of the parameters equal to $\psi_s = -0.8$, $m = 0.6$. The transient simulations have a uniform initial condition of $\psi_0 = 0$ and are run until steady state is reached. The results of these simulations are shown at three different times in Figure 3. The steady state is reached by infiltration from the top and from the channel, as can be seen in the top and middle panel. A phreatic aquifer forms at the bottom of the domain and the water table (surface at $\psi = 0$ near the bottom) moves from left to right until a regular distribution of the pressure gradients is reached.

*6.2. Steady-state results*

From Table 1 we note that our Restarted Newton BFGS algorithm provides an improvement in terms of number of iterations, irrespective on the $k_{\max}$ value and on the initial preconditioner. In almost all the runs there is also a reduction of the CPU time. The iteration number reduction appears to be monotone with $k_{\max}$, however high values of this parameter may lead to increased CPU time due to the increasing cost per iteration.

| $P_0$ | $k_{\max}$ | NLIT | LINIT | CPU |
|---|---|---|---|---|
| IC(0) | 0 | 12 | 1619 | 98.55 |
| IC(0) | 1 | 12 | 1522 | 88.30 |
| IC(0) | 2 | 12 | 1496 | 87.22 |
| IC(0) | 3 | 12 | 1475 | 88.58 |
| IC(0) | no restart | 12 | 1428 | 82.76 |
| AINV(0.05) | 0 | 12 | 4075 | 191.27 |
| AINV(0.05) | 1 | 12 | 3815 | 189.77 |
| AINV(0.05) | 3 | 12 | 3663 | 191.20 |
| AINV(0.05) | 5 | 12 | 3577 | 193.99 |
| AINV(0.02) | 0 | 12 | 2840 | 282.12 |
| AINV(0.02) | 2 | 12 | 2610 | 244.88 |
| AINV(0.02) | 5 | 12 | 2519 | 227.47 |
| AINV(0.02) | no restart | 12 | 2481 | 204.14 |

Table 1: Number of linear iterations (LINIT) and CPU time for the solution of the Picard-linearized systems with different initial preconditioners and values of the restart parameter $k_{\max}$. Steady state case.

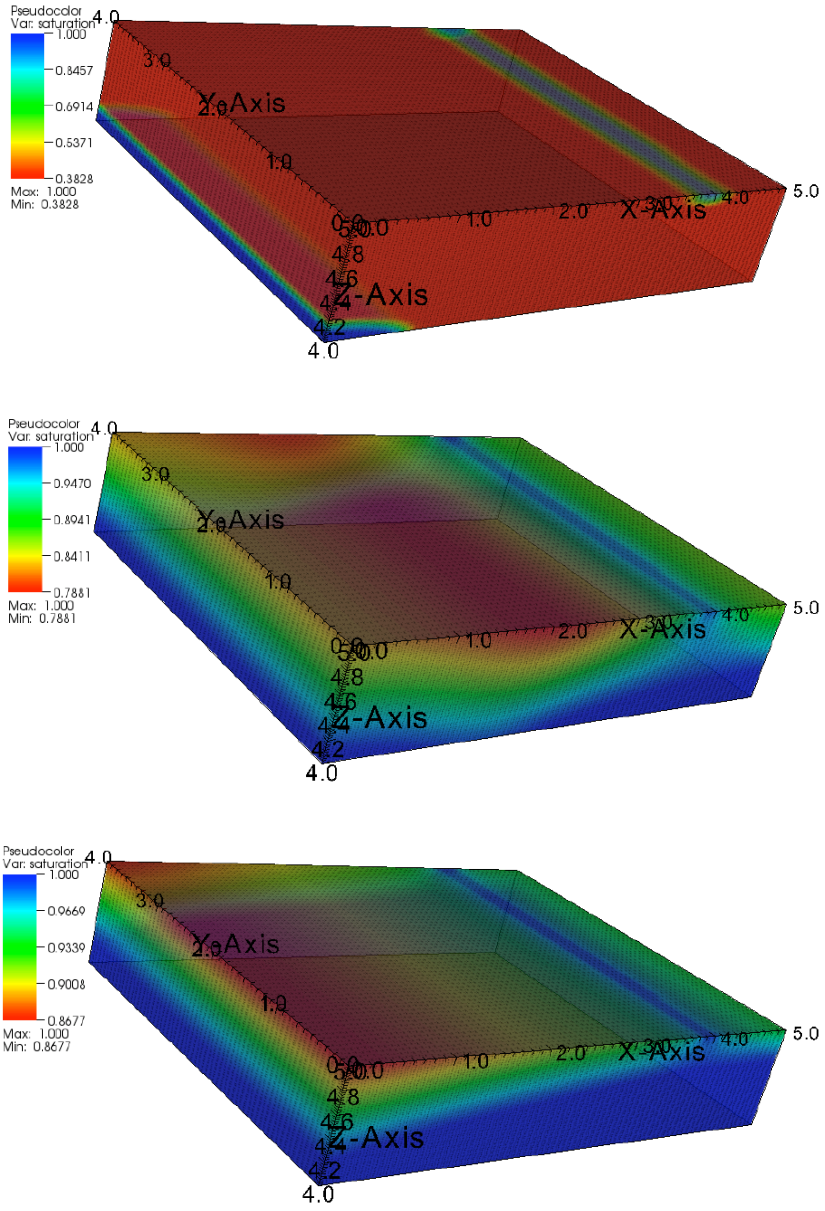


Figure 3: Results of the simulations at times $t = 3333$ seconds (top), $t = 6666$ seconds (middle), and steady state (bottom).

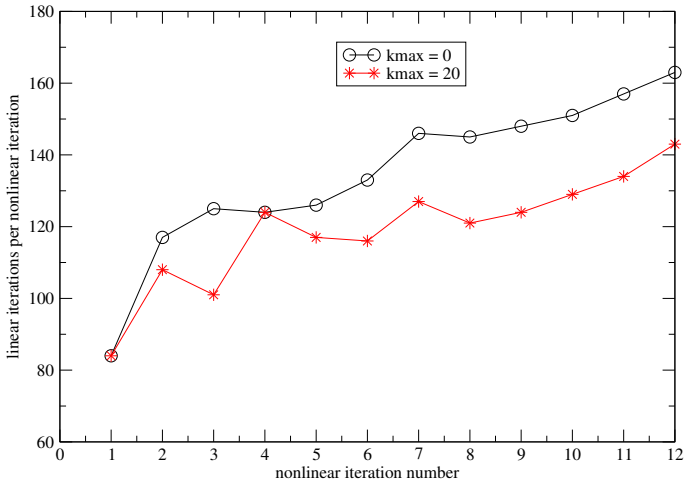


Figure 4: Iteration number for IC(0) preconditioner ($k_{\max} = 0$) and Newton BFGS preconditioner with no restart ($k_{\max} = 20$).

The advantage of the proposed preconditioner is particularly evident when the computation of the initial preconditioner is costly. This is the case of AINV(0.02) (last four rows in Table 1) where the optimal preconditioner reveals the non restarted Newton-BFGS. Here, the increasing cost per iteration is counterbalanced by the saved cost of the preconditioner computation. Finally, we note from Figure 4 that the improvement provided by BFGS update in terms of linear iterations is observed at (almost) each nonlinear iteration.

*6.3. Preliminary results for the transient case*

In Table 2 we report our first results of our preconditioner in the solution of the Picard systems in the transient case. Since the number of iterations does not change very much with $k_{\max}$ we did not report the CPU times. These preliminary results show once again a (small) improvement in terms of iteration number from $k_{\max} = 0$ (that is IC(0) computed and not updated at every nonlinear iteration) to $k_{\max} > 0$. Note that the efficiency of the BFGS update is lower with respect to the steady state case due to the well conditioning of the linearized systems. This aspect will be analyzed in the next Section 6.4.

| $P_0$ | $k_{\max}$ | timesteps | NLIT | LINIT |
|---|---|---|---|---|
| IC(0) | 0 | 22 | 151 | 12097 |
| IC(0) | 1 | 22 | 151 | 11883 |
| IC(0) | 2 | 22 | 151 | 11848 |
| IC(0) | no restart | 22 | 151 | 11822 |

Table 2: Number of linear iterations (LINIT) for the solution of the Picard-linearized systems with IC(0) as the initial preconditioner and different values of the restart parameter $k_{\max}$. Transient case.

*6.4. Condition number of the preconditioned matrices.*

To better understand the improvement obtained by the proposed acceleration, we report in Figure 5 the spectral condition number, i. e. the ratio between the largest and the smallest

eigenvalue for the preconditioned Jacobians both in the steady state case and – for a few initial timesteps – in the transient case.

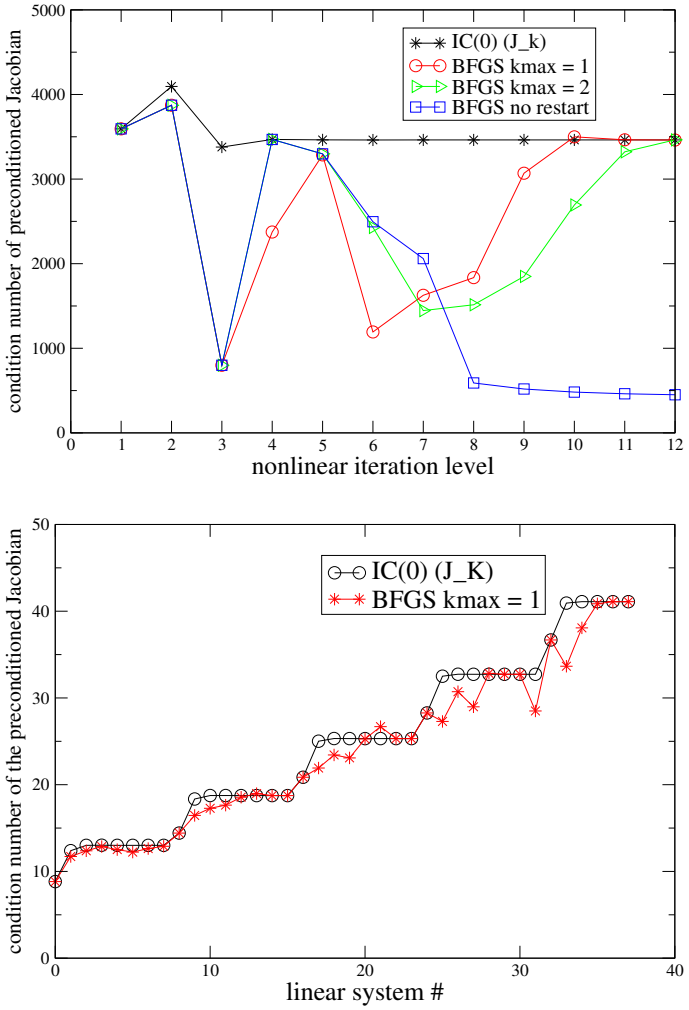


Figure 5: Spectral condition number of the preconditioned Jacobians. In the steady state case (top figure) it is computed at different nonlinear iteration levels. In the transient case (bottom figure) it is computed for the initial timesteps.

In the steady state case there is a strong reduction of the condition number provided by BFGS acceleration which can be up to a factor 4, depending on the nonlinear iteration level. In the transient case, especially for the initial time steps, where the $\Delta t$ values are smaller, there is only a slight decrease of the condition number of the preconditioned matrices, which results in a small reduction of the number of iterations as also accounted in Table 2. In fact, our algorithm adapts the timestep to the nonlinear character of the system, trying to maintain the largest possible value. To avoid unnecessary iterations, the initially small timestep is increased as time progresses. Since the condition number of the linearized system matrix is proportional to $\Delta t$, the results reveal well-conditioning at initial steps, resulting in a reduced efficiency of the BFGS update. Later timesteps show a behavior similar to the steady state, and are not reported here.

## 7. Conclusions and future developments

A rank-two update sequence of preconditioners has been proposed for accelerating the PCG method for the solution of the inner linear systems of the inexact Newton-Picard method applied to the discretized Richards equation. It has been shown that the sequence of the preconditioners is SPD and that $||I - P_k J(\boldsymbol{x}_k)||$ remains bounded if the initial vector guess $\boldsymbol{x}_0$ and $P_0$ are close enough to the exact solution and to the inverse of the Jacobian $J^{-1}(\boldsymbol{x}_0)$, respectively.

Our numerical experiments onto large size problems show that this algorithm provides an improvement of the convergence compared with computing the IC or AINV preconditioners, and this is particularly efficient for small values of the restart parameter $k_{\max}$. The proposed technique has a number of advantages on simply computing a preconditioner of $J(\boldsymbol{x}_k)$: (a) it always reduces the number of iterations, irrespective on the $k_{\max}$ value and on $P_0$; and (b) it reduces the cost of forming the preconditioner.

As a consequence, we expect that this technique, together with the one developed in [5], could be particularly effective e. g., in the interior point (IP) solution of constrained optimization problems, where linearized saddle-point Newton systems are very ill-conditioned toward the solution and the cost of the preconditioner computation may be prohibitive (see [7]).

Our future work is also aimed at using a more sophisticated preconditioner, such as the BIF algorithm, recently developed in [9], as well as more dense IC preconditioners based on fill-in and threshold, as the initial $P_0$, to provide a more important reduction in the iteration number.

## 8. References


[1] J. Bear, *Hydraulics of Groundwater*, McGraw-Hill, New York, 1979.

[2] M. Benzi, *Preconditioning techniques for large linear systems: a survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[3] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[4] L. Bergamaschi, R. Bru, and A. Martínez, *Low-rank update of preconditioners for the inexact Newton method with SPD jacobian*, Mathematical and Computer Modelling, 54 (2011), pp. 1863–1873.

[5] L. Bergamaschi, R. Bru, A. Martínez, and M. Putti, *Quasi-Newton preconditioners for the inexact Newton method*, Electronic Trans. Num. Anal., 23 (2006), pp. 76–87.

[6] L. Bergamaschi, R. Bru, A. Martínez, and M. Putti, *Quasi-Newton acceleration of ILU preconditioners for nonlinear two-phase flow equations in porous media*, Advances in Engineering Software, (2012). Published online 4 December 2010.

[7] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.

[8] L. Bergamaschi and M. Putti, *Mixed finite elements and Newton-type linearization for the solution of Richard's equation*, Int. J. Numer. Methods Engrg., 45 (1999), pp. 1025–1046.

[9] R. Bru, J. Marin, J. Mas, and M. Tuma, *Improved Balanced Incomplete Factorization*, SIAM J. on Matrix Anal. Appl., 31 (2010), pp. 2431–2452.

[10] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Num. Anal., 19 (1982), pp. 400–408.

[11] P. A. Forsyth, Y. S. Wu, and K. Pruess, *Robust numerical methods for saturated-unsaturated flow with dry initial conditions in heterogeneous media*, Adv. Water Resources, 18 (1995), pp. 25–38.

[12] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[13] C. T. Kelley, *Iterative methods for optimization*, vol. 18 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[14] C. Paniconi and M. Putti, *A comparison of Picard and Newton iteration in the numerical solution of multidimensional variably saturated flow problems*, Water Resour. Res., 30 (1994), pp. 3357–3374.

[15] C. Paniconi and E. F. Wood, *A detailed model for simulation of catchment scale subsurface hydrologic processes*, Water Resour. Res., 29 (1993), pp. 1601–1620.

[16] J. R. Philip, *Theory of infiltration*, Adv. Hydrosci., 5 (1969), pp. 215–296.

[17] P. J. Ross, *Efficient numerical methods for infiltration using Richards' equation*, Water Resour. Res., 26 (1990), pp. 279–290.

[18] M. T. van Genuchten and D. R. Nielsen, *On describing and predicting the hydraulic properties of unsaturated soils*, Ann. Geophys., 3 (1985), pp. 615–628.