Capillarity

Vol. 6, No. 2, p. 31-40, 2023

Original article

Open-source high-performance software packages for direct and inverse solving of horizontal capillary flow

Gabriel S. Gerlero[1], Claudio L. A. Berli[2], Pablo A. Kler[1,3]*

[1]*Centro de Investigación de Métodos Computacionales (CIMEC, UNL–CONICET), Colectora RN 168 km 472, Santa Fe, Argentina*

[2]*Instituto de Desarrollo Tecnológico para la Industria Química (INTEC, UNL–CONICET), Colectora RN 168 km 472, Santa Fe, Argentina*

[3]*Departamento de Ingeniería en Sistemas de Información, Facultad Regional Santa Fe, Universidad Tecnológica Nacional, Lavaisse 610, Santa Fe, Argentina*

Keywords:
Richards equation
horizontal flow
nonlinear diffusion
Boltzmann transformation
Julia language

Cited as:
Gerlero, G. S., Berli, C. L. A., Kler, P. A. Open-source high-performance software packages for direct and inverse solving of horizontal capillary flow. Capillarity, 2023, 6(2): 31-40.
https://doi.org/10.46690/capi.2023.02.02

Abstract:
This work introduces *Fronts*, a set of open-source numerical software packages for nonlinear horizontal capillary-driven flow problems in unsaturated porous media governed by the Richards equation. The software uses the Boltzmann transformation to solve such problems in semi-infinite domains. The scheme adopted by *Fronts* allows it to be faster and easier to use than other tools, and provide continuous functions for all involved fields. The software is capable of solving problems that appear in hydrology, but also in other particular domains of interest such as paper-based microfluidics. As the first known open-source implementation to adopt this approach, *Fronts* has been validated against analytical solutions as well as existing software achieving remarkable results in terms of computational costs and numerical precision, and is meant to aid the study and modeling of capillary flow. *Fronts* can be freely downloaded and installed, and offers a friendly environment for new users with its complete documentation and tutorial cases.

1. Introduction

Fluid flow in unsaturated porous media can be modeled with the Richards equation (Richards, 1931). This is a nonlinear partial differential equation that describes changes in moisture content and/or pressure from the effects of gravity and capillary action. Porous-media flow models provide closed-form expressions for the necessary constitutive relations (Sun et al., 2021). A special case of the Richards equation occurs where the flow is horizontal or the effect of gravity can be otherwise neglected-where it is also known as the moisture diffusivity equation (Bear and Cheng, 2010).

As a partial differential equation, the Richards equation is commonly solved with numerical methods based on the finite difference, finite element, or finite volume methods, paired with either fixed-point iteration (Picard's method) or Newton–Raphson method to deal with the nonlinear terms (Caviedes-Voullième et al., 2013; Lai and Ogden, 2015; List and Radu, 2016; Farthing and Ogden, 2017). Software packages that follow these schemes are widely available and used in hydrology, and range from tools designed exclusively for one-dimensional cases (Šimůnek et al., 2016) to others that extend support to arbitrary geometries (Horgue et al., 2022). However, the aforementioned nonlinear terms included in the equations to be solved imply serious computational challenges for these methods, and hence research on improved numerical schemes for these applications is still ongoing. Such research can consider one-dimensional vertical, or horizontal, geometries. In the first group, for instance, List and Radu (2016) proposed a new linearization scheme with better convergence properties; Alastal and Ababou (2019) developed a particular *Moving*

Yandy Scientific Press

*Corresponding author.
E-mail address: ggerlero@cimec.unl.edu.ar (G. S. Gerlero); cberli@santafe-conicet.gov.ar (C. L. A. Berli); kler@cimec.unl.edu.ar (P. A. Kler).

Received January 9, 2023; revised January 28, 2023; accepted February 10, 2023; available online February 15, 2023.

Table 1. Summary of different characteristics of the currently available software for solving the richards equation.

Software/algorithm	Open source	Vertical/2D/3D	Boltzmann transformation	Method	Reference
Hydrus	No(*)	Yes	No	Finite element	Šimůnek et al. (2016)
COMSOL	No	Yes	No	Finite element	Li et al. (2009)
porousMultiphaseFoam	Yes	Yes	No	Finite volume	Horgue et al. (2022)
PFLOTRAN	Yes	Yes	No	Finite volume	Hammond et al. (2014)
Pseudospectral method	Yes(**)	No	Yes	Chebyshev diff.	Mathias and Sander (2021)
Fronts	Yes	No	Yes	ODE integr.	Gerlero et al. (This work)

Notes: * 1D-only public-domain variant of this software has been discontinued, ** As implemented in the Julia version of *Fronts*. Originally published as a MATLAB script (op. cit.).

Multi-Front method for solving the Richards equation for unsaturated flow in vertical homogeneous porous columns, and Shanmugam et al. (2020) presented a weighting scheme for the computation of hydraulic conductivities that can produce more accurate results. A small snapshot of the different software currently available for solving Richards equation is reported in Table 1.

Alternatively, in the presence of horizontal flow, the Richards equation is susceptible to the Boltzmann transformation[1] (Boltzmann, 1894) . This similarity transformation reduces the nonlinear partial differential equation to an equivalent nonlinear ordinary differential equation, and can be applied to problems of horizontal flow in semi-infinite domains (Bear and Cheng, 2010). This approach has seen adoption by researchers looking for analytical solutions to problems of unsaturated flow (Philip, 1960; Prevedello et al., 2008; Su et al., 2017); however, exact solutions are limited to problems with diffusivity functions restricted to a few particular forms, excluding those currently accepted as the most accurate. More recently, the Boltzmann transformation has been applied in the formulation of semi-analytical methods for these problems (Chen and Dai, 2015; Tzimopoulos et al., 2015). Hayek (2018) presented a general approximate analytical method for the Richards equation in horizontal flow; unfortunately, as discussed in the paper, the method is incompatible with certain constitutive models (notably, the Van Genuchten model). Mathias and Sander (2021) presented a pseudospectral method for direct and inverse solving of horizontal Richards equation based on the proprietary MATLAB® software. The Boltzmann transformation has also been used in inverse problems of horizontal flow where the saturation profile can be approximated with an analytic function (Evangelides et al., 2010).

Here, it must be noted that the Boltzmann transformation can also be useful combined with standard numerical treatment (Klute, 1952; Philip, 1955; Braddock and Parlange, 1980; Andersen, 2023), resulting in a more efficient (and/or more precise) scheme against traditional numerical methods used for the Richards equation. Notably, such a scheme can obviate the need for time-stepping iterations. Furthermore, it is possible to avoid the requirement of a computational mesh as an input to the program, which makes for a solver easier to use. Moreover, when compared to an approximate analytical method, the numerical approach does not suffer from the incompatibility that appears when diffusivity function gradient tends to infinite under limit conditions for saturation as it was outlined by Hayek (2018).

This paper introduces *Fronts* (Gerlero et al., 2022a), a numerical tool for solving horizontal cases of the Richards equation with the application of the Boltzmann transformation. As such, *Fronts* is designed to handle problems where the flow occurs along a single axis and the domains may be presumed to be semi-infinite. Built with the goal of being open source, and easier to use and faster than the existing tools that can solve these problems, the software has independent implementations in two different programming languages: Python and Julia, with versions uploaded to the official package registry of each ecosystem. Both variants include the solvers that shall be described later, a set of tutorial cases, and online reference documentation that specifies their usage. The choice of environments reflects current usage trends in scientific computing, with Python being a popular environment for this type of application, while Julia is a newer language with special properties favor its adoption for numerical applications where computation times are an important factor (Bezanson et al., 2017). *Fronts* can be used both to solve problems in which the underlying hydraulic functions are known, selected from a library of built-in constitutive models, or arbitrarily provided by users; as well as inverse problems in which the parameters of an (also arbitrary) constitutive model are to be found. *Fronts* was developed with the broader community in mind. Besides its expected use in hydrology in general, a more specific application field is identified in paper-based microfluidics, where the Richards equation is now being applied in models of lateral flow assays (Schaumburg et al., 2018; Asadi et al., 2022).

On the development side, *Fronts* is open source and follows modern software development practices, notably including the

[1]The Boltzmann transformation is not compatible with gravitational terms in the general Richards equation. Nevertheless, it is possible to use the solution of a horizontal problem as starting point for the solution to a problem in which gravity is considered (Chen and Dai, 2015).

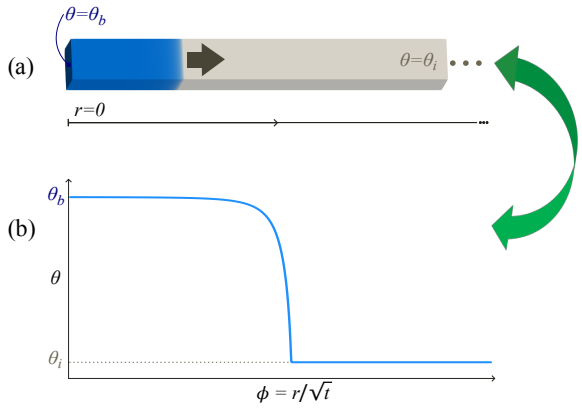


Fig. 1. (a) Initial-boundary value problem in a semi-infinite rectangular domain compatible with the Boltzmann transformation and *Fronts*, (b) transformation of the solution to the Boltzmann variable.

use of a continuous integration service with automated test suites for validation.

The paper continues with discussions of the mathematical foundations and the numerical algorithms of *Fronts*. Following that, validation of the software in terms of error calculation from well known analytical solutions is presented, as well as the discussion of some of the numerous tutorial cases that are included with the implementations. Finally, an evaluation of the capabilities of *Fronts* in solving challenging parameter estimation problems is presented. The paper finishes with concluding remarks regarding the benefits and perspectives of this new software.

2. Mathematical model and numerical method

2.1 Governing equation and Boltzmann transformation

Consider the transient nonlinear diffusion equation along a single axis **r**, which may be written as:

$$\frac{\partial \theta}{\partial t} = \nabla \cdot \left[D(\theta) \frac{\partial \theta}{\partial r} \hat{\mathbf{r}} \right] \tag{1}$$

In that expression, θ is a function of position r and time t, and D is a positive function of the range of θ. In general, θ may be called the concentration field and D the diffusivity function.

When the equation is used to model capillary flow in porous media, θ is the moisture content field, and D is known as the moisture diffusivity function, a constitutive relation that takes its expression from unsaturated flow models, with adjustable parameters that are fit to the results of certain experiments. Widely adopted constitutive relations for capillary flow are those by Brooks and Corey (1964) and Van Genuchten (1980)-implementations of which are already included in the packages-along with more recently developed models such as the LET correlations (Lomeland, 2018). Also included as part of the packages is the implementation of the novel LETd model for capillary flow, which is a LET-based simplification specially targeted for paper-based microfluidic applications (Gerlero et al., 2022b). The mathematical expressions of these models are covered in the Supplementary Material.

With an appropriate choice of the unit vector $\hat{\mathbf{r}}$, Eq. (1) is susceptible to the Boltzmann transformation[2]. The Boltzmann transformation requires the introduction of a new variable ϕ-also known as the Boltzmann variable, defined as:

$$\phi = \frac{r}{\sqrt{t}} \tag{2}$$

To apply the transformation to the partial differential equation, the partial derivatives in Eq. (1) are replaced using the following chain rules:

$$\frac{\partial}{\partial r} = \frac{\partial \phi}{\partial r} \frac{\partial}{\partial \phi} = \frac{1}{\sqrt{t}} \frac{\partial}{\partial \phi} \tag{3}$$

$$\frac{\partial}{\partial t} = \frac{\partial \phi}{\partial t} \frac{\partial}{\partial \phi} = -\frac{\phi}{2t} \frac{\partial}{\partial \phi} \tag{4}$$

After performing the substitutions, all occurrences of r and t can either be replaced with ϕ or factored out. Expanding all derivatives using the product rule, and with the final restriction that θ be a univariate function of ϕ, the following ordinary differential equation is obtained:

$$-\frac{\phi}{2} \frac{\mathrm{d}\theta}{\mathrm{d}\phi} = D(\theta) \frac{\mathrm{d}^2\theta}{\mathrm{d}\phi^2} + \frac{\mathrm{d}D}{\mathrm{d}\theta} \left(\frac{\mathrm{d}\theta}{\mathrm{d}\phi} \right)^2 \tag{5}$$

2.1.1 Initial and boundary conditions

For compatibility with the Boltzmann transformation, the problems governed by Eq. (5) are defined in semi-infinite spatial domains as it is shown in Fig. 1. Consequently, the domains will be deemed unbounded toward positive r, and the uniform initial condition:

$$\theta(r, t = 0) = \theta_i \tag{6}$$

transforms into the boundary condition at infinity for Eq. (5):

$$\lim_{\phi \to \infty} \theta = \theta_i \tag{7}$$

The previous initial condition is commonly combined with a Dirichlet boundary condition at $r = 0$, i. e.:

$$\theta(r = 0, t) = \theta_b \tag{8}$$

which by Eq. (2) transforms to:

$$\theta(\phi = 0) = \theta_b \tag{9}$$

Boundary conditions other than Eq. (8) are applicable in some particular situations and are also included within the package; these are covered in the Supplementary Material.

2.2 Reconstructing the solution

The transformed equation and initial-boundary conditions together define a two-point boundary value problem. Once a solution to this two-point problem has been found, the solution to the original problem can be reconstructed by applying the

[2]From now on, it shall be assumed that $\hat{\mathbf{r}}$ is a Cartesian unit vector. It is also possible to define $\hat{\mathbf{r}}$ to deal with some problems in cylindrical and spherical coordinates; for more details on these problems and alternative applications of the Boltzmann transformation, refer to the Supplementary Material.

Boltzmann transformation in reverse. The univariate function θ can be transformed into a function of r and t with Eq. (2) so that:

$$\theta(r,t) = \theta(\phi = \frac{r}{\sqrt{t}}) \tag{10}$$

Having the derivative $d\theta/d\phi$, the partial derivatives $\partial\theta/\partial r$ and $\partial\theta/\partial t$ can be reconstructed with Eqs. (3) and (4) respectively.

In many applications, knowing the diffusive flux is also of interest. In a problem of capillary flow, the diffusive flux measures the Darcy velocity of the fluid; as such, the diffusive flux $\mathbf{q}$ can be used as input for solving transport problems (Gerlero et al., 2022b). This quantity can be recovered as:

$$\mathbf{q} = -D(\theta)\frac{\partial\theta}{\partial r}\hat{\mathbf{r}} = -D(\theta)\frac{1}{\sqrt{t}}\frac{\mathrm{d}\theta}{\mathrm{d}\phi}\hat{\mathbf{r}} \tag{11}$$

Another quantity of interest that appears in flow problems is that of sorptivity-defined as the cumulative infiltration per square root of unit time-(Mathias and Sander, 2021), which can also be computed from the solution to the Boltzmann-transformed problem as:

$$S = -2D(\theta)\frac{\mathrm{d}\theta}{\mathrm{d}\phi} \tag{12}$$

where θ and $d\theta/d\phi$ are evaluated at the boundary by convention to obtain a scalar value for the sorptivity S.

2.3 Inverse problem

Fronts can also provide solutions to inverse problems of Eq. (1) in which the unknown is the function D. This functionality makes use of the integral expression (refer to the Supplementary Material for its derivation):

$$D(\theta) = -\frac{1}{2}\frac{\mathrm{d}\phi}{\mathrm{d}\theta}\int_{\theta_i}^{\theta}\phi(\theta)\,\mathrm{d}\theta \tag{13}$$

and may be evaluated without iteration, as long as a sufficiently continuous form of the solution-which appears here as $\phi(\theta)$-is available. In such cases, it can provide a useful $D(\theta)$ function without a dependency on known models. In real cases where the solution is known as discrete data points (with possible uncertainties) and/or under the requirement that D must conform to some known parameterized expression, the practical usefulness of this expression is diminished (Bruce and Klute, 1956), and defining an optimization problem that invokes a direct solver may be a more effective approach.

2.4 Numerical implementation

2.4.1 Direct solver

In order to solve a problem governed by the moisture diffusion equation, the first step is to convert it into a boundary value problem of Eq. (5) using the Boltzmann transformation. Then, the transformed problem is solved numerically with a shooting algorithm in combination with a root finding method.

The default solvers in both the Julia and Python implementations of *Fronts* use Radau IIA implicit numerical integrators of order 5, as implemented by the *DifferentialEquations.jl* (Rackauckas and Nie, 2017) and SciPy (Virtanen et al., 2020) libraries respectively. In practice, this algorithm of numerical integration was found to be the most accurate and robust among the available methods for a variety of tested problems.

It should be noted that derivative evaluations are required in the process of solving Eq. (5). In the first place, the derivative $\mathrm{d}D/\mathrm{d}\theta$ appears directly in such equation and thus will need evaluation. Moreover, the use of an implicit integrator imposes a requirement that derivatives of Eq. (5) must also be runtime evaluable (which at the same time implies a second differentiation of D). Derivative calculations are handled differently in each implementation of *Fronts*: in the case of Julia, the package uses forward-mode automatic differentiation (Revels et al., 2016) internally so that required derivatives are evaluated in a manner transparent to the user. However, in the Python package, the required differentiated expressions must be included as part of the implementation, usually obtained with the help of a symbolic engine (except for any function expressions provided by users at runtime, which are differentiated symbolically with the SymPy library (Meurer et al., 2017) just ahead of use). The different approaches reflect the fact that the use of automatic differentiation in Python was determined to be exceptionally expensive in terms of execution times, even after accounting for the overall speed difference that exists across the languages.

At the start of each outer iteration of the solver algorithm, a candidate value needs to be selected for the integrator to use as the free parameter (the derivative at the boundary). With the assumption that this derivative varies continuously with the initial condition, the solver uses the bisection method to select trial values from a search interval. By default, a search interval for the bisection algorithm is found with a heuristic method. This scheme has been found to be very solid in practice; its implementation in *Fronts* adds a few optimizations in order to improve robustness and avoid unnecessary computations. Notably, this rootfinding algorithm is implemented as a coroutine in Julia with the aid of the *ResumableFunctions.jl* package (Lauwens, 2017).

It is important to emphasize that the solvers do not require a starting candidate solution or even a grid of points to be provided by the user as an input. Rather, the discretization of the solution is fully controlled by the automatic step selection of the numerical integrator, which constitutes an advantage in terms of usability and performance.

2.4.2 Inverse solver

The inverse solver takes the input function in discrete form, as values of θ on a finite set of points of the Boltzmann variable. Such information can be experimental data of moisture content as a function of time and space (Villarreal et al., 2019). It first constructs a continuous function ϕ of θ by interpolating the input data as a cubic spline using the PCHIP method (Fritsch and Butland, 1984). Subsequently, the diffusivity function D is defined by applying the integral expression of Eq. (13).

Whenever the returned diffusivity function is invoked, it computes the values of the required expressions by evaluating the interpolating spline accordingly-making use of its integral and derivatives when required, so as to conform to the same

requirements that the software imposes on built-in and user-defined diffusivity functions. As a result, an inverse solver invocation can be expected to be faster than a direct solver call, as it does not rely on an iterative algorithm. However, it bears the disadvantages of the approach mentioned in Section 2.3.

2.4.3 Parameter estimation

Alternatively, an optimization-based parameter estimation strategy can be employed to find values for parameters (including initial and boundary values) for particular expressions of the diffusivity D, as provided for instance by any established (or newly developed) model for capillary flow. This scheme implies the definition of an objective function that itself invokes the direct solver and compares against the desired results. The objective function can then be passed to any optimization package to obtain estimates for the free parameters. Given the substantial performance advantage, the Julia version of *Fronts* is preferred for this type of usage. Indeed, to facilitate this mode of usage, an optimization support module is included in that implementation, which assists the user in creating an appropriate objective function for a particular model, set of free parameters, solver configuration, and arbitrary target data points with possible uncertainties. Anticipating the fact that the objective functions will be repeatedly invoked by any chosen optimizer, computational efficiency is of particular relevance in the implementation of this functionality; notably, objective functions constructed from this module are able to fit constant factors in D as part of an inner intermediate stage, which avoids otherwise unnecessary solver invocations (Gerlero et al., 2022) in estimating this common type of parameter.

3. Results and discussion

In this section, different cases as validation or application examples are discussed. It is relevant to note that many more example cases are included as part of the packages—due to space constrains, it will cover a subset of those that are considered the most significant for showcasing the broad range of applications of *Fronts*. All reported execution times were measured with the Julia version of the software running on an M1 MacBook Air (Apple Inc., Cupertino, Calif., USA) notebook computer. In order to run *Fronts*, the single requirement is a standard installation of either Julia (version 1.6 or newer) or Python (version 3.7 or newer).

3.1 Software validation

In order to validate the software implementation, two particular cases that admit exact analytical solutions—due to the special forms of their diffusivity functions (Philip, 1960)—were solved with *Fronts*. The results show both remarkable numerical precision and low computational costs.

3.1.1 Case 1

For the first case, the proposed diffusivity function is:

$$D(\theta) = \frac{m\theta^m}{2}\left(1 - \frac{\theta^m}{m+1}\right), m > 0. \quad (14)$$

where the initial and boundary conditions are $\theta_i = 0$ and $\theta_b = 1$ respectively.

An unusual feature of this case is that, although the wetting front is finite, its derivative at $\phi = 1$ is discontinuous and infinite when approaching from the left. As was demonstrated by Philip (1960), the exact solution for this case is:

$$\theta = (1-\phi)^{\frac{1}{m}}, \phi \in [0,1] \quad (15)$$

For the solution of this case three different values of m are considered, i. e. $m = \{1,2,5\}$. Fig. 2 gathers the results of this case for the different values of m. Fig. 2(a) shows the comparison of the analytical solution from Eq. (15) and the solution computed with *Fronts*. Figs. 2(b) and 2(c) show the behavior of the error (mean deviation from analytical solution) for the moisture content and sorptivity as a function of solver tolerance (absolute tolerance for θ_i). In both cases, a strong linear dependence can be inferred up to a limit value of the tolerance that was found to be around 10^{-4}; for lower values of such tolerance, the errors behave asymptotically. Finally, Fig. 2(d) shows the number of iterations required by solver to obtain a solution within the prescribed tolerance; the log-linear trend shows the clear advantage of the adopted computational implementation. For a fixed tolerance of 10^{-3} (default value in the software), the measured execution times were $m = 1 : 0.4012$ ms, $m = 2 : 0.9641$ ms, $m = 5 : 2.096$ ms; which are comparable to similar results reported by Mathias and Sander (2021) for a pseudospectral method implemented on a proprietary platform.

3.1.2 Case 2

In this second case also proposed by Philip (1960) and used for validation of the implementation, the diffusivity function is written as:

$$D(\theta) = \frac{m}{2(m+1)}\left[(1-\theta)^{m-1} - (1-\theta)^{2m}\right], m > 0 \quad (16)$$

where the initial and boundary conditions are once again $\theta_i = 0$ and $\theta_b = 1$.

The particular feature of this case is that the solution θ exhibits an infinite derivative at $\phi = 0$. The exact solution for this case is:

$$\theta = 1 - \phi^{\frac{1}{m}}, \phi \in [0,1] \quad (17)$$

For the solution of this second case, four different values of m are considered; i. e. $m = \{1,2,5,10\}$. Fig. 3 shows the results of the second validation case for the different values of m. Fig. 3(a) shows the comparison of the analytical solution calculated from Eq. (17) and the numerical solution produced by *Fronts*. Once more, a remarkable agreement between the numerical and analytical solution is observed. Figs. 3(b) and 3(c) show the behavior of the mean error for the moisture content and percent error for sorptivity as a function of the solver tolerance. In both instances, a strong linear dependence can be inferred up to a limit tolerance value that, in this case, shows a dependency on m for the moisture content error, where a higher m coincides with a greater irreducible error. As for sorptivity, the stagnation error is higher than in Case 1, as was also observed with other methods (Parlange and Braddock

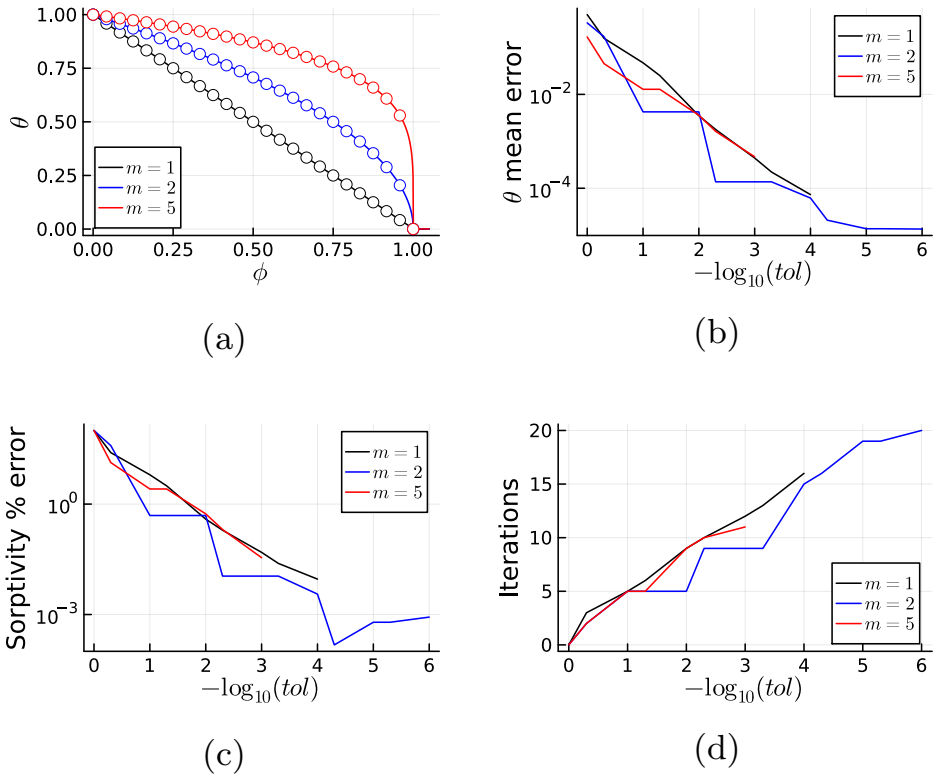


Fig. 2. Validation results of Case 1. (a) Comparison of numerical (circles) and analytical (solid line) solutions, (b) and (c) moisture content (mean absolute) and sorptivity (percent) errors as function of solver tolerance and (d) number of solver iterations as a function of solver tolerance.

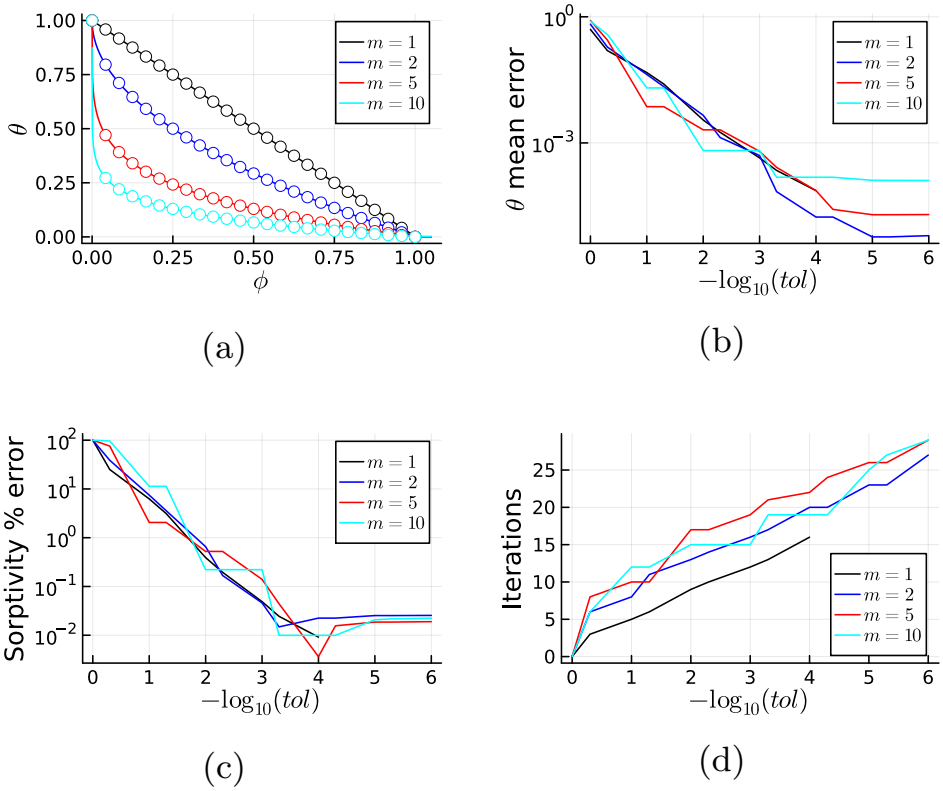


Fig. 3. Validation results of Case 2. (a) Comparison of numerical (circles) and analytical (solid line) solutions, (b) and (c) moisture content (mean absolute) and sorptivity (percent) errors (respectively) as function of solver tolerance and (d) number of solver iterations as a function of solver tolerance.

Table 2. Initial-boundary conditions and model parameters for the *Grenoble sand* application example (Fuentes et al., 1992).

Parameter		Value
Initial condition (asymptotic)	θ_i	0.3120
Boundary condition (asymptotic)	θ_b	0
Van Genuchten parameters	K_s	15.37 (cm/h)
	α	0.0432 (/cm)
	m	0.5096
	l	0.5
	θ_r, θ_s	$\theta_r = \theta_i$, $\theta_s = \theta_b$

(1980); Parlange et al. (1994), as evaluated by Mathias and Sander (2021))-yet notably not with Mathias and Sander's (2021) own method. Finally, Fig. 3(d) shows the number of iterations required to solve the case following the prescribed tolerance. As also occurred with Case 1, the trend here is log-linear, showing once again the advantages of the solver implementation. For the default tolerance of 10^{-3}, the measured computation times were $m = 1 : 0.4147$ ms, $m = 2 : 1.778$ ms, $m = 5 : 2.725$ ms, $m = 10 : 1.877$ ms, which are also comparable to those previously reported.

3.2 Application examples

Two of the eight tutorial cases that are included with the implementations of *Fronts* are discussed here. The first is an archetypal example of capillary imbibition with severe ill-conditioning that is known to not be solvable by other methods based on the Boltzmann transformation. Finally, a parameter estimation example based on previously published experimental data of imbibition in filter paper is presented.

3.2.1 Direct solving example

For the first application example, a case of capillary imbibition into a porous material with properties described with the Van Genuchten model was selected. It is important to note that no analytical solutions exist for this widely adopted flow model in general nor in this particular example (Hayek, 2018).

This case adopts the parameters for *Grenoble sand* (Fuentes et al., 1992) as listed in Table 2 along with the initial and boundary conditions, which are set to θ_r (i.e., completely dry medium) and θ_s (fully saturated) respectively. Given that the Van Genuchten diffusivity increases without limit as θ approaches θ_s and evaluates to zero when θ is equal to θ_r, the problem of infiltration into a fully dry medium with this model is intrinsically ill-conditioned and both the initial and boundary conditions have to be treated as asymptotic. Indeed, this case was previously presented by Hayek (2018) as one that could not be correctly solved by application of his approximation method. For its part, *Fronts* is able to produce an accurate solution for this problem within the default tolerance for the initial value and a very small (10^{-7}) absolute tolerance for the boundary value.

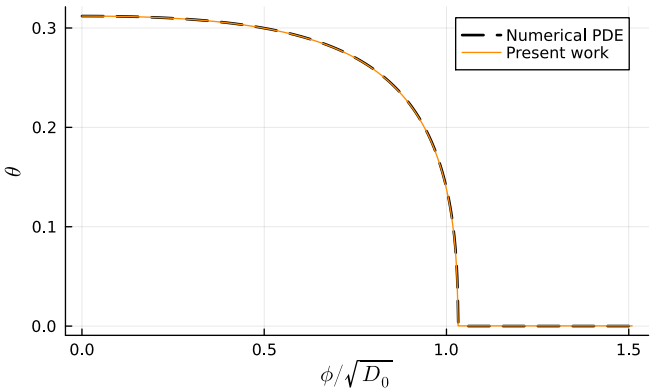


Fig. 4. Results of the first application example. The solid line is the solution produced by *Fronts*, while the dashed line shows the solution obtained with other software (*porousMultiphaseFoam*) that uses a classical numerical approach for the partial differential equation.

Fig. 4 shows the solution produced by *Fronts* for this case, with the x-axis showing the Boltzmann variable normalized by the constant $\sqrt{D_0}$, where $D_0 = (1-m)K_s/(\alpha m \theta_s)$. For comparison, also included is the solution obtained for the same problem with a classical numerical approach that considers the full partial differential equation, as implemented in the open-source *porousMultiphaseFoam* toolbox that uses the finite-volume method (Horgue et al., 2022). Even with this scheme, a tolerance for the initial condition needs to be introduced in order to avoid the ill-conditioning at the limit value. With these allowances, it can be observed that both tools predict equivalent solutions to this problem, with *Fronts* being superior in terms of execution times (more than ten thousand-fold difference) due to its more specialized algorithm.

3.2.2 Parameter estimation example

For the final case to cover, experimental data from Gerlero et al. (2022b) was used, which studied the capillary imbibition process in Whatman No. 1 paper, a material that is broadly accepted as the most important substrate for paper-based microfluidics. The dataset contains 141 datapoints of moisture content at different values of the Boltzmann variable, each contemplating the experimental uncertainty, that represent the observed wetting profile for this material.

The example cases of parameter estimation included with the package consider two different flow models: Van Genuchten and LETd. It is important to note that this two models were selected for this substrate due to their known potential for describing the imbibition process in this substrate with few parameters; yet a user can arbitrarily choose any of the other implemented models (described in the Supplementary Material) or propose a different one based on their own needs.

The cases follow a similar strategy than in Gerlero et al. (2022b), but here adopting the new parameter estimation support module provided by *Fronts*. That module is used to

Table 3. Results of parameter estimation using *Fronts* and the experimental data.

Model	Parameter	Search interval	Value
Van Genuchten	K_s/α	(constant factor)	2.105×10^{-6} (m^2/s)
	m	$[0.001, 0.999]$	0.8861
	l	$[-50, 50]$	2.331
	θ_r	$[0, \theta_i]$	0.005 623
LETd	D_{wt}	(constant factor)	1.004×10^{-3} (m^2/s)
	L	$[0, 10]$	1.356
	E	$[0, 10^5]$	10 010
	T	$[0, 10]$	1.224
	θ_r	$[0, \theta_i]$	0.006 25

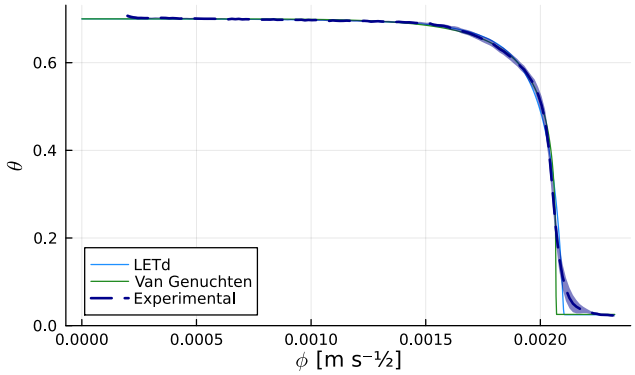
generate an appropriate cost function that is subsequently minimized with an optimizer of the differential evolution family as implemented in the *BlackBoxOptim.jl* (Feldt, 2022) package for Julia.

Table 3 lists the search intervals and found values of the free parameters for each of the models after allowing the optimization to run for 60 seconds on a single thread, with the fitness measured with the reduced chi-square statistic (χ^2_ν) (Taylor, 1997), which considers the uncertainty in the experimental data (the constant factor parameters fitted by the parameter estimation module as described in Section 2.4.3). Fig. 5(a) compares the solutions found by parameter estimation. Fig. 5(b) shows the diffusivity functions corresponding to the fitted models and the diffusivity predicted by the inverse solver (Section 2.4.2). Both flow models show good agreement with the experimental data in terms of the moisture content profiles, with errors $\chi^2_\nu = 1.6$ and 0.9 for the Van Genuchten and LET models respectively. For their part, the moisture diffusivity curves deviate from each other at low values of θ. The diffusivities estimated via the inverse method are purportedly more representative of the experimental data in this region due to not being constrained by any particular model. Nevertheless, the large discrepancies in the diffusivities translate to only minor differences in the predicted profiles, which validates the implementation of parameter estimation that considers the predicted profiles in lieu of what would be a far less expensive fitting of the diffusivity models to the results of the inverse solver.

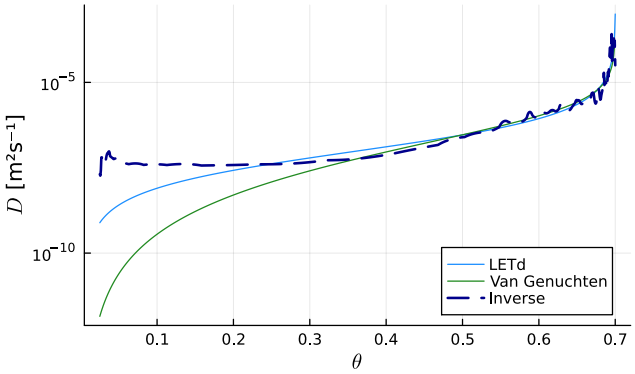
4. Conclusions

This paper introduced *Fronts*, a new software tool for solving nonlinear diffusion problems numerically. To our knowledge, it is the first instance of open-source software that adopts a scheme based on the Boltzmann transformation to solve such problems. Distributed as Julia and Python packages, the implementations includes tutorial cases and reference documentation for the entire set of features. *Fronts* is openly available under the MIT License (Julia) and BSD 3-clause license (Python).

The implementations have been validated in two well-known and challenging cases with analytical solutions. The



(a)



(b)

Fig. 5. Results of the parameter estimation examples considering the Van Genuchten and LETd models. (a) Results of fitting the experimental data with either model (shaded area behind the experimental curve represents the uncertainty in the data) and (b) comparison of diffusivities predicted by the fitted models with the results of calling the inverse solver with the experimental data.

robustness and capabilities of *Fronts* have also been demonstrated in solving a case with no possible solution when using other methods. At the same time, *Fronts* was shown to also be simpler to use: the main solver has no parameters of numerical origin that need to be set or tuned when attempting to solve a case. These facts are once more restated as clear advantages of the approach adopted in *Fronts*.

Direct (predictive) use of the toolbox notwithstanding, *Fronts* has proven to be useful in solving parameter estimation problems through optimization runs-made practicable due its speed, included functionality and ease of automation, as was demonstrated in the imbibition-based characterization of Whatman No. 1 paper.

Moreover, *Fronts* can easily apply arbitrary diffusivity functions defined by users in infiltration problems. The fact that the partial derivatives and diffusive flux of the solution are obtained in continuous form is also notable when reconstructing velocity fields. Furthermore, there is also potential for the toolbox to help with coupled transport problems, in which

the diffusive flux from a problem compatible with *Fronts* forms an advection velocity in a different problem. While advection problems obviously fall outside of the scope of the toolbox, having access to *Fronts* in this scenario means that the diffusion problem needs to be solved just once, and the continuous solution passed on to a different solver that itself only needs to deal with the (usually linear) transport problem.

With the release of this work, a positive impact is foreseen in the field of fluid flow in porous media at different scales, ranging from applications in hydrology to paper-based microfluidics, where *Fronts* is expected to help in the study of capillary flow and the development of new processes, techniques and devices.

Acknowledgements

This work was supported by CONICET, Agencia I+D+i (No. PICT-2018-02920) and UTN (No. PID8132, No. PID8685), Argentina. The authors acknowledge Prof. Raúl Urteaga and PhD. Andrés R. Valdez for fruitful discussions on the topic of parameter estimation in capillary flow in porous media.

Supplementary file

https://doi.org/10.46690/capi.2023.02.02

Conflict of interest

The authors declare no competing interest.

References

Alastal, K., Ababou, R. Moving multi-front (MMF): A generalized Green-Ampt approach for vertical unsaturated ows. Journal of Hydrology, 2019, 579: 124184.

Andersen, P. Ø. Insights from Boltzmann transformation in solving 1D counter-current spontaneous imbibition at early and late time. Advances in Geo-Energy Research, 2023, 7(3): 164-175.

Asadi, H., Pourjafar-Chelikdani, M., Khabazi, N. P., et al. Quasi-steady imbibition of physiological liquids in paper-based microfluidic kits: Effect of shear-thinning. Physics of Fluids, 2022, 34(12): 123111.

Bear, J., Cheng, A. H. D. Modeling groundwater flow and contaminant transport, in Theory and Applications of Transport in Porous Media, edited by Bear, J., Springer, Dordrecht, pp. 89-103, 2010.

Bezanson, J., Edelman, A., Karpinski, S., et al. Julia: A fresh approach to numerical computing. SIAM Review, 2017, 59(1): 65-98.

Braddock, R. D., Parlange, J. Y. Some accurate numerical solutions of the soil-water diffusion equation. Soil Science Society of America Journal, 1980, 44(3): 656-658.

Brooks, R. H., Corey, A. T. Hydraulic properties of porous media. Fort Collins, Colorado State University, 1964.

Bruce, R. R., Klute, A. The measurement of soil moisture diffusivity. Soil Science Society of America Journal, 1956, 20(4): 458-462.

Boltzmann, L. To integrate the diffusion equation with variable diffusion coefficients. Annalen der Physik, 1894, 289(13): 959-964. (in German)

Caviedes-Voullième, D., García-Navarro, P., Murillo, J. Verification, conservation, stability and efficiency of a finite volume method for the 1D Richards equation. Journal of Hydrology, 2013, 480: 69-84.

Chen, X., Dai, Y. An approximate analytical solution of Richards' equation. International Journal of Nonlinear Sciences and Numerical Simulation, 2015, 16(5): 239-247.

Evangelides, C., Arampatzis, G., Tzimopoulos, C. Estimation of soil moisture profile and diffusivity using simple laboratory procedures. Soil Science, 2010, 175(3): 118-127.

Farthing, M. W., Ogden, F. L. Numerical solution of Richards' equation: A review of advances and challenges. Soil Science Society of America Journal, 2017, 81(6): 1257-1269.

Feldt, R. BlackBoxOptim.jl. GitHub 2022.

Fritsch, F. N., Butland, J. A method for constructing local monotone piecewise cubic interpolants. SIAM Journal on Scientific and Statistical Computing, 1984, 5(2): 300-304.

Fuentes, C., Haverkamp, R., Parlange, J. Y. Parameter constraints on closed-form soilwater relationships. Journal of Hydrology, 1992, 134(1-4): 117-142.

Gerlero, G. S., Kler, P. A., Berli, C. L. A. Fronts.jl. GitHub 2022a.

Gerlero, G. S., Valdez, A. R., Urteaga, R., et al. Validity of capillary imbibition models in paper-based microfluidic applications. Transport in Porous Media, 2022b, 141(2): 359-378.

Hammond, G. E., Lichtner, P. C., Mills, R. T. Evaluating the performance of parallel subsurface simulators: An illustrative example with PFLOTRAN. Water Resources Research, 2014, 50(1): 208-228.

Hayek, M. An efficient analytical model for horizontal infiltration in soils. Journal of Hydrology, 2018, 564: 1120-1132.

Horgue, P., Renard, F., Gerlero, G. S., et al. porousMultiphaseFoam v2107: An open-source tool for modeling saturated/unsaturated water flows and solute transfers at watershed scale. Computer Physics Communications, 2022, 273: 108278.

Klute, A. A numerical method for solving the flow equation for water in unsaturated materials. Soil Science, 1952, 73(2): 105-116.

Lai, W., Ogden, F. L. A mass-conservative finite volume predictor-corrector solution of the 1D Richards' equation. Journal of Hydrology, 2015, 523: 119-127.

Lauwens, B. ResumableFunctions: C# sharp style generators for Julia. Journal of Open Source Software, 2017, 2(18): 400.

Li, Q., Ito, K., Wu, Z., et al. COMSOL Multiphysics: A novel approach to ground water modeling. Groundwater, 2009, 47(4): 480-487.

List, F., Radu, F. A. A study on iterative methods for solving Richards' equation. Computational Geosciences, 2016, 20: 341-353.

Lomeland, F. Overview of the LET family of versatile correlations for flow functions. Paper SCA2018-056 Presented at International Symposium of the Society of Core Analysts, Trondheim, Norway, 27-30 August, 2018.

Mathias, S. A., Sander, G. C. Pseudospectral methods provide fast and accurate solutions for the horizontal infiltration equation. Journal of Hydrology, 2021, 598: 126407.

Meurer, A., Smith, C. P., Paprocki, M., et al. SymPy: Symbolic computing in Python. PeerJ Computer Science, 2017, 3: e103.

Parlange, J. Y., Barry, D. A., Parlange, M. B., et al. Sorptivity calculation for arbitrary diffusivity. Transport in Porous Media, 1994, 15: 197-208.

Parlange, J. Y., Braddock, R. D. An application of Brutsaert's and optimization techniques to the nonlinear diffusion equation: The influence of tailing. Soil Science, 1980, 129(3): 145-149.

Philip, J. R. Numerical solution of equations of the diffusion type with diffusivity concentration-dependent. Transactions of the Faraday Society, 1955, 51: 885-892.

Philip, J. R. General method of exact solution of the concentration-dependent diffusion equation. Australian Journal of Physics, 1960, 13: 1.

Prevedello, C. L., Loyola, J. M., Reichardt, K., et al. New analytic solution of Boltzmann transform for horizontal water infiltration into sand. Vadose Zone Journal, 2008, 7(4): 1170-1177.

Rackauckas, C., Nie, Q. Differentialequations. jl-a performant and feature-rich ecosystem for solving differential equations in Julia. Journal of Open Research Software, 2017, 5(1).

Revels, J., Lubin, M., Papamarkou, T. Forward-mode automatic differentiation in Julia. arXiv preprint, 2016, 1607: 07892.

Richards, L. A. Capillary conduction of liquids through porous mediums. Physics, 1931, 1(5): 318-333.

Schaumburg, F., Urteaga, R., Kler, P. A., et al. Design keys for paper-based concentration gradient generators. Journal of Chromatography A, 2018, 1561: 83-91.

Shanmugam, M., Kumar, G. S., Narasimhan, B., et al. Effective saturation-based weighting for interblock hydraulic conductivity in unsaturated zone soil water flow modelling using one-dimensional vertical finite-difference model. Journal of Hydroinformatics, 2020, 22(2): 423-439.

Šimůnek, J., Van Genuchten, M. T., Šejna, M. Recent developments and applications of the HYDRUS computer software packages. Vadose Zone Journal, 2016, 15(7): vzj2016.04.0033.

Su, L., Wang, Q., Qin, X., et al. Analytical solution of a one-dimensional horizontal-absorption equation based on the Brooks-Corey model. Soil Science Society of America Journal, 2017, 81(3): 439-449.

Sun, J., Li, Z., Furtado, F., et al. A microfluidic study of transient flow states in permeable media using fluorescent particle image velocimetry. Capillarity, 2021, 4(4): 76-86.

Taylor, J. R. Introduction To Error Analysis: The Study of Uncertainties in Physical Measurements. New York, USA, University Science Books Mill Valley, 1997.

Tzimopoulos, C., Evangelides, C., Arampatzis, G. Explicit approximate analytical solution of the horizontal diffusion equation. Soil Science, 2015, 180(2): 47-53.

Van Genuchten, M. T. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Science Society of America journal, 1980, 44(5): 892-898.

Villarreal, R., Lozano, L. A., Melani, E. M., et al. Diffusivity and sorptivity determination at different soil water contents from horizontal infiltration. Geoderma, 2019, 338: 88-96.

Virtanen, P., Gommers, R., Oliphant, T. E., et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods, 2020, 17(3): 261-272.