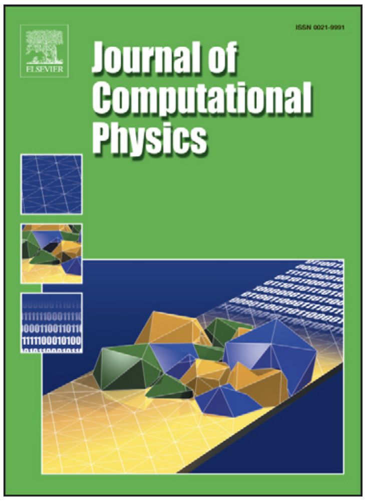

Simulating Finger Phenomena in Porous Media with a Moving Finite Element Method

Guanghui Hu, Paul Andries Zegeling

PII: S0021-9991(11)00056-8
DOI: 10.1016/j.jcp.2011.01.031
Reference: YJCPH 3432

To appear in: *Journal of Computational Physics*

Received Date: 16 August 2010
Revised Date: 11 January 2011
Accepted Date: 22 January 2011

Please cite this article as: G. Hu, P.A. Zegeling, Simulating Finger Phenomena in Porous Media with a Moving Finite Element Method, *Journal of Computational Physics* (2011), doi: 10.1016/j.jcp.2011.01.031

Simulating Finger Phenomena in Porous Media with a Moving Finite Element Method

Guanghui Hu[*,a], Paul Andries Zegeling[b]

[a]Department of Mathematics, Michigan State University, M. I., U. S.
[b]Department of Mathematics, Utrecht University, Utrecht, the Netherlands.

Abstract

The non-equilibrium Richards equation is solved using a moving finite element method in this paper. The governing equation is discretized spatially with a standard finite element method, and temporally with second-order Runge-Kutta schemes. A strategy of the mesh movement is based on the work by Li et al. [R. Li, T. Tang and P. W. Zhang, A moving mesh finite element algorithm for singular problems in two and three space dimensions. J. Comput. Phys., 177(2002), pp. 365-393]. A Beckett and Mackenzie type monitor function is adopted. To obtain high quality meshes around the wetting front, a smoothing method which is based on the diffusive mechanism is used. With the moving mesh technique, high mesh quality and high numerical accuracy are obtained successfully. The numerical convergence and the advantage of the algorithm are demonstrated by a series of numerical experiments.

Key words: Porous medium, finger phenomenon, non-equilibrium RE, gravity driven flow, moving finite element method

1. Introduction

In gravity-driven infiltration into initially dry, homogeneous soil, the resulting pattern often takes the form of preferential flow paths (fingers), which have been consistently observed in laboratory and field experiments for nearly half a century[6, 14, 16]. The experimental and mathematical study of gravity-driven unstable flows over the past three decades has led to the conclusion that unstable flow can occur over a wide range of field conditions[11]. Among many mathematical models used for modeling fingering phenomenon, the Richards equation (RE) is one of the most typical models[24]. Many works have been done following the RE model. In [22], Otto gave the non-linear stability analysis of Richards equation, and pointed out that the RE model can not produce instability of a wetting front in homogeneous porous media. In [11], Egorov et al extended the stability analysis of [22] to the heterogeneous porous media, and proposed a possible modification of the Richards equation, say, the so-called non-equilibrium Richards equation (NERE) model. Recently, in [6], another modification of the Richards equation is given by adding a non-local energy term, which leads to excellent numerical results. In this paper, we focus on the numerical simulations of the NERE model.

The stability analysis of the NERE model is given in [11], where a low-frequency instability criterion (LFC) condition is proposed to describe the stability of NERE model when there is a low frequency perturbation on the wetting front. With appropriate parameters, the NERE model will generate non-monotonic distribution of pressure. The infiltrating flows governed by the NERE can become unstable for conditions where the flow profiles is sufficiently non-monotonic [21].

Numerical simulation is a very important tool for investigating finger phenomenon. Many excellent works have been done on numerical simulations of finger phenomenon, see, e.g., [6, 9, 10, 25]. According to the

[*]Corresponding author
Email addresses: ghhu@math.msu.edu (Guanghui Hu), P.A.Zegeling@uu.nl (Paul Andries Zegeling)

appearance of the physical solution (the finger), an ideal mesh used in the simulation should be able to capture both the hold-back-pile-up phenomenon and each finger on the wetting front. For configurations such as small initial saturation in the dry region, the wetting front could be very sharp, which means that the mesh size should be sufficiently small to resolve such sharp profile successfully. Consequently, we will need a uniformly dense mesh to take care of those interesting physical phenomenon, because the sharp wetting front can move with the time evolution. Note that at any fixed time, a sufficiently dense mesh is only needed around the wetting front. In the region far away from the wetting front, we actually just need a relatively coarse mesh, because there is almost no solution variation. Based on the above observations, the usage of a mesh redistribution technique in each time step may optimize the algorithm, say, the number of mesh points and CPU time may be saved. In this paper, an adaptive moving mesh method is used for this goal.

The moving mesh method used in this paper is originally proposed in[19, 20], and has been widely used in many applications, see, e.g., [27, 23, 28, 29, 30]. People can find out a summary of the theory and applications of moving mesh method in [28]. This moving strategy is based on harmonic mappings, and the mesh quality can partially be controlled by certain indicators relevant to the solution singularities, it is the so-called monitor function. After moving the mesh, solutions can be updated from the old mesh to the new mesh by solving an ODE system, which is derived from the assumption that the surface of solution is unchanged during the mesh redistribution. The main advantages of this moving mesh method contain the following two aspects. The first one is that the grid points on the boundary can move together with the internal grid points in a unified way. For problems with a singularity on the boundary, such technique can improve the numerical accuracy significantly, see, e.g, [17]. Another advantage is that the mesh redistribution part is independent of numerical schemes for solving the governing equations. The implementation of the moving mesh method is realized in a C++ library AFEPack[18].

The monitor function is very important for the implementation of the algorithm. Generally, a gradient-based monitor function is used,

$$m = \sqrt{\varepsilon + \mid \nabla S \mid^2}, \tag{1}$$

where $\varepsilon > 0$ is a parameter which controls the adaptivity, and S denotes the saturation in this paper. It is found that an appropriate selection of ε highly depends on the problems. For different parameters in the governing equations, the value of ε may change dramatically. To enhance the robustness of the algorithm, a monitor function which is similar to a Beckett and Mackenzie [12, 13] type monitor function is used, which replaces ε with a time-dependent function $\mathcal{M}(t)$. We follow [3, 8] to choose the form of $\mathcal{M}(t)$. With this time-dependent function, the floor on the monitor function is adjusted automatically in proportion to the measure of the solution gradient. From the numerical simulations, we can see that such monitor function works very well, and mesh movement is not sensitive to the variation of the parameters in the monitor function. The Beckett and Mackenzie type monitor function is also adopted in [8], and impressive numerical results are presented there.

To avoid a highly irregular mesh around the wetting front, a smoothing-step for the monitor function is necessary. However, certain parameters in the simulations which have sufficient a small initial saturation in the dry region will cause very sharp wetting fronts, and the smoothing-step proposed in [19, 20] may not handle a good mesh quality any more. Furthermore, note that the initial saturated region is quite small compared to the whole physical domain, we need a much more powerful smoothing method to cluster sufficient mesh grids around this region. Fortunately, the smoothing-step proposed by Wang et al.[30] which is based on the diffusive mechanism can help to improve the mesh redistribution. With such a smoothing-step, the mesh points which are far away from the finger are clustered successfully around the wetting front, and the variation of the mesh nearby the singularity becomes sufficiently smooth, which guarantees the quality of the numerical solution. We will see these advantages in the numerical simulations.

The rest of this paper is organized as follows. In section 2, the RE model and the NERE model are reviewed, and the stability of the NERE model is briefly outlined. In section 3, we first give the spatial and temporal discretization of the NERE model, and then describe the moving mesh strategy. The choice of the monitor function and the smoothing method are also described in this section. The numerical experiments are carried out and analyzed in section 4. Conclusions are drawn in the final section.

2. Models

2.1. The Richards equation

The conventional Richards equation for the flow of water in unsaturated porous media may be written in dimensionless form as:

$$\frac{\partial S}{\partial t} = \nabla \cdot K(S)\nabla p + \frac{\partial}{\partial z}K(S), \tag{2}$$

$$p = \mathcal{P}(S), \tag{3}$$

where S is the effective saturation ($0 \leq S \leq 1$), p is the water pressure, K is the hydraulic conductivity. The function $\mathcal{P}$ is the equilibrium pressure being a function of S. Equation (3) is the standard approach for modeling the capillary pressure, and is often called static capillary pressure.

The above system can be reduced to one equation with S being a primary variable by introducing the diffusivity function $D(S) = K(S)\mathcal{P}'(S)$ (prime means the first derivative with respect to the saturation S):

$$\frac{\partial S}{\partial t} = \nabla \cdot D(S)\nabla S + \frac{\partial}{\partial z}K(S). \tag{4}$$

Since the Richards equation has been proved unconditionally stable[11], it is not suitable to be used for simulating finger phenomenon. In [15, 11], based on the stability analysis of the Richards equation, a new model is proposed, namely the non-equilibrium Richards equation (NERE). Egorov et al. analyzed this new model and gave the conditional stability results. In the following two subsections, we briefly summarize the NERE model and its stability analysis.

2.2. Non-Equilibrium Richard equation

If we keep the mass balance equation (2) and replace the equilibrium relation (3) with a kinetic equation such as

$$\mathcal{F}(S, p, \dot{S}, \dot{p}, \ddot{S}, \ddot{p}, \cdots) = 0, \tag{5}$$

where the variables $\dot{S}$ and $\ddot{S}$ mean the first and second order derivatives of the saturation with respect to the time respectively, the so-called non-equilibrium Richards equation is obtained.

Note that (5) is a general form. In numerical simulations, a specific expression[15, 11] is suggested

$$\tau\dot{S} = p - \mathcal{P}(S), \tag{6}$$

where τ is a positive constant.

Substituting (6) into (2) gives

$$\begin{aligned}
\frac{\partial S}{\partial t} &= \nabla \cdot (K(S)\nabla(\tau\dot{S} + \mathcal{P}(S)) + \frac{\partial}{\partial z}K(S) \\
&= \nabla \cdot (K(S)\nabla(\tau\dot{S}) + \nabla \cdot (K(S)\nabla\mathcal{P}(S)) + \frac{\partial}{\partial z}K(S) \\
&= \nabla \cdot (K(S)\mathcal{P}'(S)\nabla S) + \frac{\partial}{\partial z}K(S) + \tau\nabla \cdot \left(K(S)\nabla\frac{\partial S}{\partial t}\right).
\end{aligned}$$

Similar to the Richards equation, we can get NERE with saturation as the primary variable by introducing the diffusivity function $D(S) = K(S)\mathcal{P}'(S)$ (prime means the first derivative with respect to the saturation S)

$$\frac{\partial S}{\partial t} = \nabla \cdot (D(S)\nabla S) + \frac{\partial}{\partial z}K(S) + \tau\nabla \cdot \left(K(S)\nabla\frac{\partial S}{\partial t}\right). \tag{7}$$

If $\tau = 0$, (7) becomes the standard Richards equation. When τ is big enough, the hold-back-pile-up phenomenon can be observed around the wetting front, which is a distinctive feature of fingered flows. The hold-back-pile-up phenomenon is very important in modeling finger phenomenon, which can be shown in the next subsection.

2.3. The stability of NERE model

For the detailed stability analysis of the NERE model, we refer [11] and references therein. In this subsection, we only briefly introduce the Low-Frequency instability Criterion (LFC) for the NERE model.

The LFC condition is obtained based on the comparison of following two forms of perturbed solution of the NERE model. First, the perturbed traveling wave solutions for Eqs. (2) and (5) can be written as

$$S = S_0(\xi) + \epsilon \mathcal{S}(\xi) e^{(i(\omega_x x + \omega_y y) - kt)} + O(\epsilon^2), \tag{8}$$

$$p = p_0(\xi) + \epsilon P(\xi) e^{(i(\omega_x x + \omega_y y) - kt)} + O(\epsilon^2), \tag{9}$$

where $S_0(\xi)$ and $p_0(\xi)$ are the basic traveling solutions, $\epsilon > 0$ represents the size of perturbation, $\xi = z - vt$ the traveling wave coordinate, ω ($\omega^2 = \omega_x^2 + \omega_y^2$) is the wave number with ω_x and ω_y are the angular frequencies with respect to x and y respectively, and

$$v = \frac{K(S_+) - K(S_-)}{S_+ - S_-} \tag{10}$$

the velocity of the wetting front with S_+ the initial saturation behind the wetting front and S_- the initial saturation ahead of the wetting front.

Since the traveling wave solution for the NERE model is invariant with regard to arbitrary shift ϵ, the form $(S, p) = (S_0(\xi + \epsilon), p_0(\xi + \epsilon))$ can also be viewed as one possible perturbed solution for the NERE model. By comparing the Taylor expansion of form $S_0(\xi + \epsilon)$ and $p_0(\xi + \epsilon)$ with system (8) and (9) at $k = 0$, $\omega_x = 0$ and $\omega_y = 0$, we know that the eigenfunctions corresponding to $k_0 = 0$ are S_0' and p_0', and k_0 can be expanded in power series of ω^2 as

$$k_0 = 0 + b\omega^2 + \cdots . \tag{11}$$

The value of b plays a very important role since it determines the stability of NERE model with low frequency perturbation. If b is negative, from (8) and (9) we know that the perturbations will grow with the time evolution, and that means the model may exhibit unstable features.

It is pointed out in [11] that the value of b is determined by the following equation

$$b = \frac{\displaystyle\int_{-\infty}^{+\infty} K(S_0) p_0' d\xi}{S_0(+\infty) - S_0(-\infty)}. \tag{12}$$

Note that $S_0(+\infty) > S_0(-\infty)$ in all of our simulations. Consequently, if $\int_{-\infty}^{+\infty} K(S_0) p_0' d\xi < 0$, the traveling wave solution of NERE model (2) and (5) is unstable. This is the so called Low-Frequency instability criterion (LFC)[11].

We follow the suggestion in [11] to choose parameters $K(S) = S^\alpha$ with $\alpha > 1$, and $D(S) = S^\beta$ with $\beta > 0$. So it is always true that $K(S) > 0$ since $S \in (0, 1)$. For the Richards equation, it is well known that the pressure increases with the saturation S from $-\infty$ at $S = 0$ to a limiting value at $S = 1$, which means that p' is always bigger than 0. Consequently, it is also always true that $b > 0$, which explains why the Richards equations are unconditionally stable with infinitesimal perturbations.

For the NERE model, if the parameter τ in (7) is sufficiently large, NERE will generate a non-monotonic distribution of the saturation nearby the wetting front. Fig. 1depicts the saturation with the following choice for the functions and the parameters: $D(S) = S^2$, $K(S) = S^2$, $S_+ = 0.5$, $S_- = 0.1$, and different τ: $\tau = 0.1$, $\tau = 0.5$ and $\tau = 1.0$. When τ is small ($\tau = 0.1$), the saturation varies monotonically, see Fig. 7 (solid line). With the increment of τ, the variations become non-monotonic (dashed lines), and so does the distribution of the pressure. This is called the hold-back-pile-up phenomenon. From Fig. 7, we can see that bigger τ causes the intenser oscillations after the wetting front. When the flow profile is sufficiently non-monotonic, the NERE becomes unstable. Consequently, the LFC predicts a transition from the stability to instability of solutions with increase of τ [11].

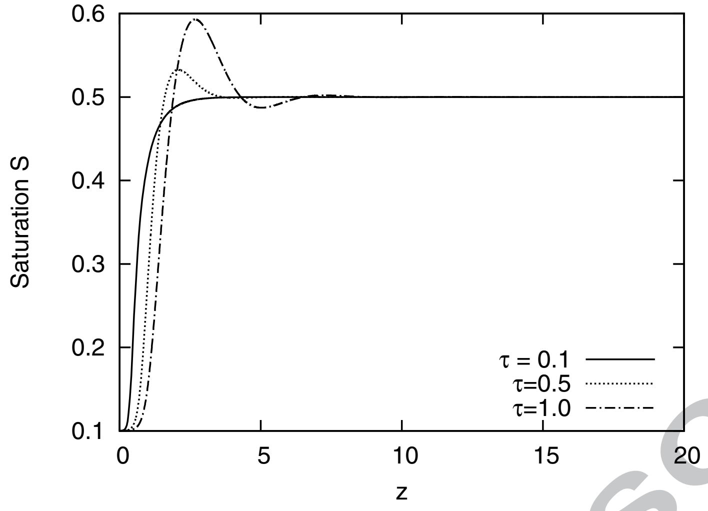

Figure 1: The saturation profiles obtained with parameters $D(S) = S^2$, $K(S) = S^2$ and different τ: $\tau = 0.1$, $\tau = 0.5$ and $\tau = 1.0$.

3. The Moving Finite Element Method for NERE Model

In this section, we will describe a numerical procedure based on moving finite element method to solve the NERE model (7). The standard finite element methods are used to discretize the governing equation spatially, then the second-order Runge-Kutta schemes are used for the temporal discretization. For resolving the hold-back-pile-up phenomenon and the sharp wetting-front successfully, the moving mesh methods which is based on the harmonic mapping are also introduced in the algorithm to redistribute the mesh points reasonably.

3.1. The spatial and temporal discretizations of NERE model

Suppose the physical domain is $\Omega \in \mathbb{R}^2$, and the triangulation of Ω is $\mathcal{T}$ with $\mathcal{T}_i$ as its elements and $\vec{X}_i$ as its nodes. The piecewise linear finite element space is denoted by V_h corresponding to $\mathcal{T}$. Then the finite element approximation of unknown S in (7) can be written as

$$S_h = \sum_{i=1}^{N_T} S_i N_i(\mathcal{T}),$$

where $\{S_i\}_{i=1}^{N_T}$ is the coefficient for variable S in (7) with N_T the dimension of V_h, and $\{N_i(\mathcal{T})\}_{i=1}^{N_T}$ the finite element basis for V_h.

For convenience, S is also used to denote the coefficient. Based on the above assumptions, the discretization formulation for (7) takes the form

$$M_S \frac{\partial S}{\partial t} = -D(S)\mathcal{K}_S S + F_S - \tau K(S)\mathcal{K}_S \frac{\partial S}{\partial t}, \tag{13}$$

where M_S and $\mathcal{K}_S$ are matrices, whose entries are

$$M_S(i,j) = \int_\Omega N_i(\mathcal{T})N_j(\mathcal{T})dxdz,$$

$$\mathcal{K}_S(i,j) = \int_\Omega \nabla N_i(\mathcal{T}) \cdot \nabla N_j(\mathcal{T})dxdz,$$

and F_S is a vector whose i-th entry is

$$(F_S)_i = \int_\Omega \frac{\partial K(S)}{\partial z} N_i(\mathcal{T}) dxdz, \qquad 1 \le i \le N_T.$$

For the time discretization, it is known that explicit schemes are efficient. However, when explicit schemes are used, the stability and the high numerical accuracy of the algorithm are guaranteed only with sufficient small length of time-step. In the simulations of this paper, we use a second-order Runge-Kutta method to discretize the temporal terms, and treat the first term of the right hand side of (13) implicitly. Let Δt be the length of time step, and suppose the solution $S^{(n)}$ on time level (n) is known, we use the following two steps to calculate the solution $S^{(n+1)}$ on the next time level $(n+1)$:

Step 1:

$$\frac{S^{(1)} - S^{(n)}}{\Delta t/2}\left(M_S + \tau K(S^{(n)})\mathcal{K}_S\right) + D(S^{(n)})\mathcal{K}_S S^{(1)} = F_S(S^{(n)}), \tag{14}$$

Step 2:

$$\frac{S^{(n+1)} - S^{(n)}}{\Delta t}\left(M_S + \tau K(S^{(1)})\mathcal{K}_S\right) + D(S^{(1)})\mathcal{K}_S S^{(n+1)} = F_S(S^{(1)}). \tag{15}$$

In the simulations, the wetting front of the fingers will be very sharp with appropriate parameters. Consequently, very dense mesh is always needed around the wetting front to resolve these fingers successfully during the whole simulation. However, efficiency of the algorithm will be significantly slowed down if the global dense mesh is used, because the region of wetting front is very small compared with the whole domain. So the adaptive techniques become necessary for optimizing the algorithm.

There are several adaptive techniques such as h-adaptive methods which locally refine and coarsen the mesh, r-adaptive methods which redistribute the mesh points while keeping the amount of mesh points unchanged, p-adaptive methods which enrich the degree of freedom locally. In this paper, we focus on the r-adaptive method, and follow [19, 20] to provide moving mesh strategy.

3.2. The moving mesh strategy

Our mesh redistribution strategy is based on the harmonic mappings. The mesh mapping between physical domain Ω with local coordinate $\vec{x}$ and computational domain Ω_c with local coordinate $\vec{\xi}$ is generated by a variation approach. Specifically, the mesh mapping is provided by the minimizer of a functional which has the following form

$$E(\vec{\xi}) = \frac{1}{2}\sum_k \int_\Omega \frac{1}{\omega}\sum_i (\frac{\partial \xi^k}{\partial x^i})^2 d\vec{x}, \tag{16}$$

where the positive function ω is a weighted function depending on the physical solution to be adapted, and the so-called monitor function can be introduced as $G = 1/\omega$. The solution of the above optimization problem is given by the following Euler-Lagrange equation

$$\sum_i \frac{\partial}{\partial x^i}\left(G\frac{\partial \xi^k}{\partial x^i}\right) = 0. \tag{17}$$

The solution of (17) is taken as the mapping between the computational domain and the physical domain.

In [19], the following procedure is used to move the mesh and to redistribute the numerical solutions. First, we get the initial mesh T_c with $\mathcal{A}$ as its nodes in the logical domain by solving the Poisson equation

$$\begin{cases} \Delta_x \xi &= 0, \quad \vec{x} \in \Omega, \\ \xi \mid_{\partial\Omega} &= \xi_b. \end{cases} \tag{18}$$

Now suppose we have obtained the solution $U_i = \vec{u}(X_i)$ on the current nodes X_i at the time step $t = t_n$. Then we get the new location of nodes X_i^* and the new solution U^* on the new nodes using the following three steps.

1. Obtain the error of mesh points in the logical domain. First, we solve the following generalized Poisson equation
$$\frac{\partial}{\partial x^i}(G^{ij}\frac{\partial \xi^k}{\partial x^j}) = 0 \tag{19}$$
together with the same boundary condition to (18) to obtain the new logical mesh T^{new} with $\mathcal{A}^{new}$ as its nodes. Then the quantity
$$\delta\mathcal{A} = \mathcal{A} - \mathcal{A}^{new} \tag{20}$$
can be used to calculate the movement of nodes in the physical domain in the next step;
2. Obtain the movement of physical nodes. By solving the following system on each element $\mathcal{T}_i$ with $X_{\mathcal{T}_{i,j}}$, $0 \le j \le 2$ as its nodes in the physical domain,
$$\begin{bmatrix} A^{new,1}_{\mathcal{T}_{i,1}} - A^{new,1}_{\mathcal{T}_{i,0}} & A^{new,1}_{\mathcal{T}_{i,2}} - A^{new,1}_{\mathcal{T}_{i,0}} \\ A^{new,2}_{\mathcal{T}_{i,1}} - A^{new,2}_{\mathcal{T}_{i,0}} & A^{new,2}_{\mathcal{T}_{i,2}} - A^{new,2}_{\mathcal{T}_{i,0}} \end{bmatrix} \begin{bmatrix} \frac{\partial x^1}{\partial \xi^1} & \frac{\partial x^1}{\partial \xi^2} \\ \frac{\partial x^2}{\partial \xi^1} & \frac{\partial x^2}{\partial \xi^2} \end{bmatrix} = \begin{bmatrix} X^1_{\mathcal{T}_{i,1}} - X^1_{\mathcal{T}_{i,0}} & X^1_{\mathcal{T}_{i,2}} - X^1_{\mathcal{T}_{i,0}} \\ X^2_{\mathcal{T}_{i,1}} - X^2_{\mathcal{T}_{i,0}} & X^2_{\mathcal{T}_{i,2}} - X^2_{\mathcal{T}_{i,0}} \end{bmatrix}, \tag{21}$$
we can get the $\partial\vec{x}/\partial\xi$ in the element $\mathcal{T}_i$. Then the weighted average error of mesh points in the physical domain is given by
$$\delta X_j = \frac{\sum_{\mathcal{T}_i} \mid \mathcal{T}_i \mid \frac{\partial\vec{x}}{\partial\xi} \mid_{\mathcal{T}_i} \delta\mathcal{A}_j}{\sum_{\mathcal{T}_i} \mid \mathcal{T}_i \mid}, \tag{22}$$
where $\mid \mathcal{T}_i \mid$ is the volume of element $\mathcal{T}_i$. Finally, we get the new mesh on the physical domain with X^{new} as its nodes
$$X^{new} = X + \tilde{\tau}\delta X$$
, where $\tilde{\tau}$ is a positive parameter and is used to avoid the intersection of new grid points. We follow [19] to choose $\tilde{\tau} = \min(0.5/ \mid\mid \delta\mathcal{A} \mid\mid_2, 0.618)$.
3. Update the solution on the new mesh. The method we used to update the solution is based on the assumption that the surface of solution on each time step will be kept unchanged. For the detail of the method, we refer to [19] and references therein.

By using the above procedure to redistribute the grid points, the singularity of the numerical solution inside the domain can be resolved effectively. However, such procedure can not be used to handle the singularity of the solution which appears on the boundary of the domain. To give the reasonable distribution of grid points on the boundary, several methods have been proposed. For example, a one-dimensional problem can be solved on the boundary to give the distribution, or the grid points on the boundary can move according to the movement of the inner grid point which are adjacent to themselves. In this paper, based on the properties of the problem, we also need that the grid points on the boundary move together with points inside the domain. To achieve this aim, we adopt the strategy which is presented in [20].

The framework which is proposed in [20] is very similar to the above moving procedure. To move all the grid points in the domain in an unified way, the initial logical mesh T_c is given by solving the following optimization problem
$$\begin{cases} \min \sum_k \int_\Omega \sum_i \left(\frac{\partial \xi^k}{\partial x^i}\right)^2 d\vec{x} \\ \text{s.t. } \xi|_{\partial\Omega} = \xi_b \in \mathbf{K}, \end{cases} \tag{23}$$
where $\mathbf{K} = \{\xi_b \in C^0(\partial\Omega)|\xi_b : \partial\Omega \to \partial\Omega_c; \xi_{b|\Lambda_i}$is a linear segment of strictly increasing$\}$ denotes a mapping set from $\partial\Omega$ to $\partial\Omega_c$. Note that different from ξ_b in (18) which is known, the ξ_b in above optimization problem is unknown in the same way as the interior points. To obtain the error in (20), now the following optimization problem is solved instead of (19)
$$\begin{cases} \min \sum_k \int_\Omega G^{ij}\frac{\partial \xi^k}{\partial x^i}\frac{\partial \xi^k}{\partial x^j} d\vec{x} \\ \text{s.t. } \xi|_{\partial\Omega} = \xi_b \in \mathbf{K}. \end{cases} \tag{24}$$

By using the above two optimization problems instead of original problems, the grid points in the whole domain can move in an unified way, which significantly improves the mesh quality for problems which have singularity on the boundary of the domain. In this paper, we use such method to redistribute grid points in each time step. In the implementation, the harmonic mapping is obtained by using a iterative method. For details of this method, we refer [20] and references therein.

Monitor Function

The monitor function is very important in the implementation of moving mesh methods. It depends on (gradients of) the solutions, and people generally use the standard arclength-type monitor function

$$m = \sqrt{\varepsilon + \mid \nabla S \mid^2}, \tag{25}$$

where ε is a positive constant. With a small ε, the high adaptivity is achieved. Conversely, there is almost no adaptivity when ε is sufficiently big. The monitor function (25) has been widely used. Unfortunately, the appropriate selection of ε highly depends on the scaling of the variables in the problem. For different problems, even for different parameters of the same problem, dramatic change may happen on the selection of ε, which causes the difficulties on the usage of monitor function (25).

In the simulation of this paper, we use the following monitor function

$$m = \sqrt{\tilde{\delta}\mathcal{M}(t) + \mid \nabla S \mid^{\alpha}}, \tag{26}$$

where α and $\tilde{\delta}$ are positive constant, and the time dependent function

$$\mathcal{M}(t) = \frac{1}{\mid \Omega \mid} \int_{\Omega} \mid \nabla S \mid^{\alpha} d\Omega,$$

where Ω is the physical domain and $\mid \Omega \mid$ is its area. Such monitor function is similar to the BM type monitor function[12, 13, 31]. In our simulations, $\alpha = 0.5$ and $\tilde{\delta} \in (0, 1)$ are suggested. With these choices, the floor on the BM monitor matrix is adjusted automatically in proportion to the measure of the (smoothed) solution gradient[3]. Compared with monitor function (25), the monitor function (26) works very well with almost one set of fixed parameters. As our numerical experience, $\alpha = 0.5$, $\tilde{\delta} = 1$ can work well for almost all numerical simulations. One can adjust the adaptivity of the algorithm by changing the value of $\tilde{\delta}$.

Smoothing Step

To avoid a very distorted mesh around the wetting front of fingers, some smoothing steps for the monitor function should be implemented. Many smoothing methods have been presented, [4, 5, 26]. In our simulation, we use the smoothing method proposed by Li et al[19]. In the algorithm, each element has a monitor value, say, a constant. First, we interpolate such monitor function from a piecewise constant to a piecewise linear function

$$(\pi_h m)|_{\vec{X}_i} = \frac{\sum_{\vec{X}_i \in \mathcal{T}_i} m|_{\mathcal{T}_i} \mid \mathcal{T}_i \mid}{\sum_{\vec{X}_i \in \mathcal{T}_i} \mid \mathcal{T}_i \mid}, \tag{27}$$

where $\mathcal{T}_i$ is a element and $\vec{X}_i$ is its nodes. Then we project this piecewise linear function back to the piecewise constant

$$m|_{\mathcal{T}_i} = \frac{1}{n+1} \sum_{\vec{X}_i \in \mathcal{T}_i} (\pi_h m)|_{\vec{X}_i \in \mathcal{T}_i}, \tag{28}$$

where $n = 2$ is the dimension of the physical domain Ω. By using the smoothing step (27) and (28) several times, the mesh quality around the wetting front is improved.

As we mentioned before, compared with the whole physical domain Ω, the initial saturated region is very small, see Fig. 2. For certain parameters such as small saturation value in the dry region, the wetting front of fingers is very sharp, so a large amount of grid points is needed around the wetting front. A straightforward method is to use very small $\tilde{\delta}$ in (26) to cluster sufficient grid points. However, if $\tilde{\delta}$ is too small, the mesh

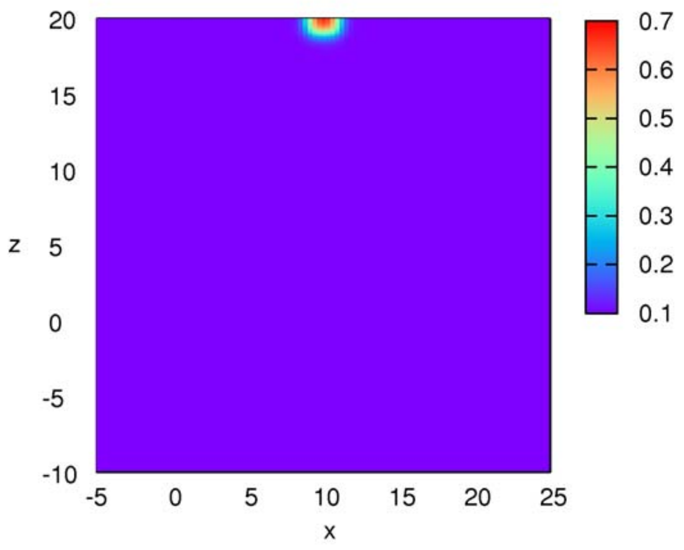

Figure 2: The computational domain and the initial configuration of a single finger. The saturation varies from 0.1 to 0.7.

quality around the wetting front becomes bad even if the smoothing steps (27) and (28) are used, see Fig. 3 (top). To fix this problem, a simple way is to use much denser mesh, then initial fingers can be resolved successfully. But note that there is still a large amount of mesh grids in the region far away from the wetting front. For the consideration of the efficiency of the algorithm, we want to keep the number of grid points unchanged, and drag those grid points towards the wetting front. To achieve this aim, the strategy proposed by Wang et al. [30] will be used. For the details of this strategy, we refer to [30] and references therein. Now we only give its implementation for our problems.

In monitor function (26), we introduce a new variable $\tilde{S}$, and the new monitor function can be read as

$$m = \sqrt{\tilde{\delta}\mathcal{M}(t) + \mid \nabla S \mid^{\alpha} + \tilde{S}}. \tag{29}$$

$\tilde{S}$ is obtained by solving the following problem

$$(1 - \mu\Delta)\tilde{S} = \mid \nabla S \mid^{\alpha}, \tag{30}$$

where μ is a positive parameter which depends on the size of Ω.

With the monitor function (29), the mesh grids which are far away from the wetting front can be moved to the wetting front successfully. Furthermore, the variation of the mesh around the wetting front becomes smooth, which guarantees the quality of the numerical solutions, see Fig. 3(bottom).

In fact, the usage of the monitor function (29) slows down the efficiency of the algorithm, because an extra diffusive problem has to be solved. However, we only need a rough solution of problem (29). So with the help of multi-grid methods, the increment of CPU time is not significant. Moreover, the monitor function (29) allows us to use much coarser mesh than that when (26) is used to solve the NERE model. In other words, the monitor function (29) can significantly improves the mesh quality around the singularity, with insignificant affection on the efficiency of the algorithm.

So far, we have stated all details about the algorithm, the flow chart of the algorithm can be summarized as the following

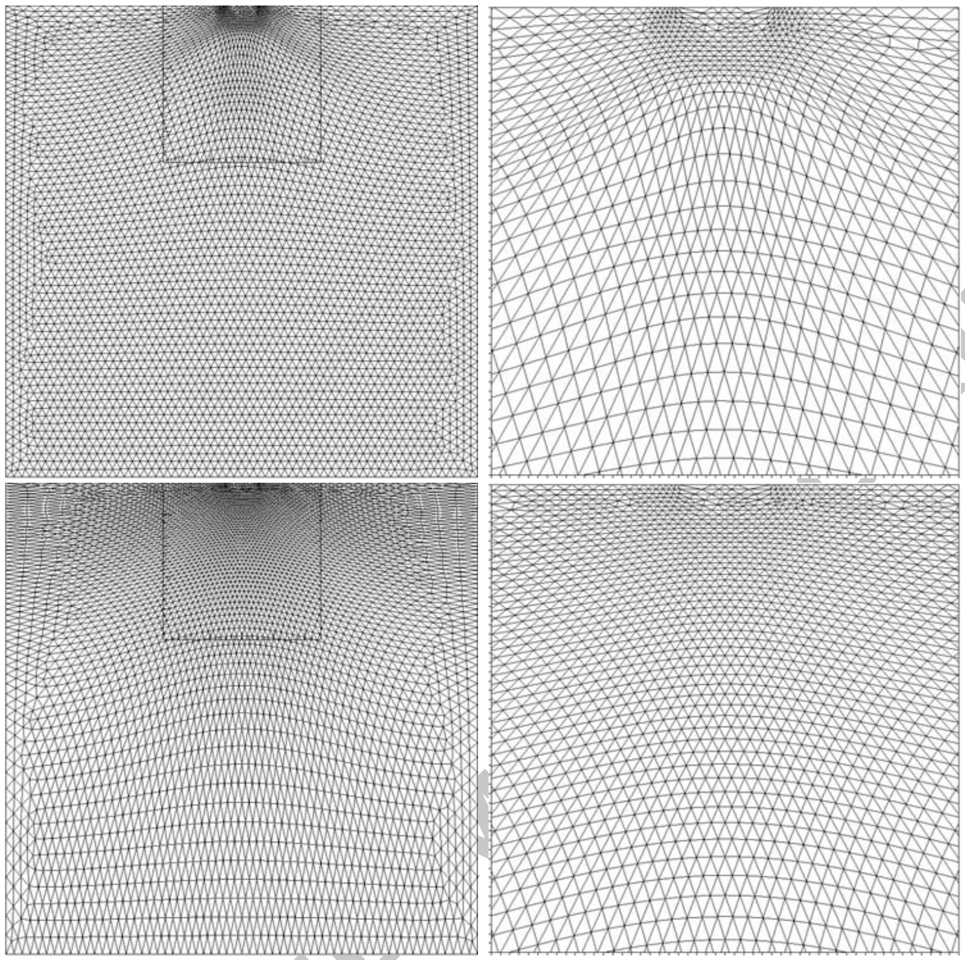
Figure 3: The initial mesh for the simulation with initial condition which is shown in Fig. 2. Top: the monitor function (26) is used, and $\delta = 0.08$; Bottom: the monitor function (29) is used with the same parameter δ.

```
Algorithm 1: Moving FEM for NERE model
1. Generate initial mesh and set the initial status for $S$;
2. Solve (14) and (15);
3. Move the mesh using the strategy proposed in Subsection 3.2;
4. Let $t = t + \Delta t$, if $t < T$, goto step 2; else stop.
```

4. Numerical results

In this section, plenty of numerical experiments will be presented. First, the numerical convergence of the moving finite element method will be demonstrated. For the 1-D case, pseudo-analytical solutions of the NERE model can be obtained, the numerical convergence can be observed easily from the the comparison between the numerical results and the pseudo-analytical one.

Then we show a simulation with one finger occurring in the spatial domain. With certain parameters and a small initial saturation, the edge of the finger will be very sharp. In this case, a very dense mesh is needed around the wetting front to resolve the sharp edge, otherwise there will be numerical oscillations. As we can see from the numerical results, our moving mesh strategy will cluster sufficient mesh points around the edge

Figure 4: The computational region

By substituting

$$S(z,t) = u(z - vt) = u(\xi) \tag{31}$$

into (7), we get

$$-v\frac{\partial u}{\partial \xi} = \frac{\partial}{\partial \xi}\left(D(u)\frac{\partial u}{\partial \xi}\right) - \frac{\partial}{\partial \xi}K(u) - \tau\frac{\partial}{\partial \xi}\left(K(u)v\frac{\partial^2 u}{\partial \xi^2}\right). \tag{32}$$

Let $u(+\infty) = 0.5$, $u(-\infty) = 0.001$, the velocity of traveling wave is $v = \dfrac{u^2(+\infty) - u^2(-\infty)}{u(+\infty) - u(-\infty)} = 0.501$. Integrating (32) in the both sides, we get the following ODE

$$-u(\xi)(v - u(\xi)) = 0.4u'(\xi) - \tau v u^2(\xi)u''(\xi) - 0.0005 \tag{33}$$

Let $\psi = u'$, (33) can be written as a Lienard type of system of two equations

$$\left\{\begin{array}{lcl} u' & = & \psi \\ \psi' & = & \dfrac{u(0.501 - u) + 0.4\psi - 0.0005}{0.501\tau u^2} \end{array}\right. \tag{34}$$

We solve the ODE system (34) using the LSODE solver, which is a ODE solver in the OCTAVE[1] software. The Fig. 5 shows the solution of traveling wave and its representation in the phase plane.

In the numerical simulation with moving finite element method, the following function is used as the initial condition

$$S_{init} = 0.2495 * \tanh(z - 97) + 0.2505. \tag{35}$$

The parameters are the same as in the earlier 1D simulation. Fig. 6 shows the differences between the pseudo-analytical solution and numerical solutions in the phase plane. The numerical results are obtained at

	Initial mesh size	CPU time(seconds)
Fixed Mesh	$\Delta h = 0.5$	88.79
Moving Mesh	$\Delta h = 1.0$	56.80
Moving Mesh	$\Delta h = 0.5$	313.22

Table 1: Comparison of the CPU time(seconds) between the fixed mesh case and the moving mesh case.

$t = 100$. Computations with the moving mesh method are performed on three successively refined meshes. As we can see from the figures, the finer the mesh, the higher the accuracy of the numerical solutions we get, which obviously shows the numerical convergence of the moving finite element method. It is worth saying that with the moving mesh strategy, the quality of solutions with an initial mesh size $\Delta h = 1$ is almost the same as the one when using a fixed mesh with a mesh size $\Delta h = 0.5$. The solution with the moving mesh strategy when $\Delta h = 0.5$ almost coincided with the pseudo-analytical solution: not only the main wetting curve is covered by numerical solutions, but also the primary and secondary curves are also covered very well.

Fig. 7 shows the numerical solution (top) around the wetting front, and its mesh distribution (bottom). It is obvious that the mesh density around the wetting front is much bigger than the region far away from it, which means that the moving mesh strategy proposed in this paper works very well.

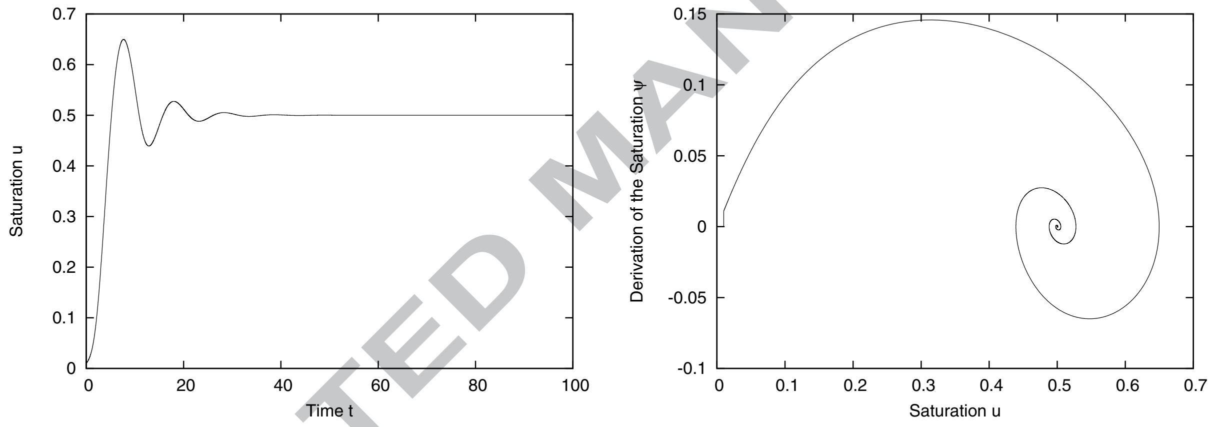

Figure 5: Pseudo-analytical solution of the ODE system 34. Left: the wave solution. Right: the solution in the phase plane. $K(S) = S^2$, $D(S) = 0.4$, $\tau = 10$

For the moving mesh method, besides solving of the governing equation, we still need to implement the mesh movement and the solution update. Consequently, more CPU time is needed compared with fixed mesh case which has the same number of grid points in the domain. However, it must be mentioned that the increment of CPU time is not significant because of the usage of the multigrid solver in solving (23), (24) and (30). Note that we only need a rough solution from (30), the equation is just solved for a couple of time steps.

The advantage of moving mesh method is that the grid points can be redistributed towards singularity of the solution. That means to achieve the same quality of the numerical solution, less grid points are needed for the moving mesh method. This results in a decrease of the CPU time. To demonstrate this, Table 1 gives a comparison of CPU time between the fixed mesh case and the moving mesh case. The CPU time shown in the table is the CPU seconds which are used to simulate the time evolution of the governing equation from $t = 15$ to $t = 20$. As we can see, with the same initial mesh size $\Delta h = 0.5$, the moving mesh method (313.22 CPU seconds) takes more CPU seconds than the fixed mesh (88.79 CPU seconds). But the solution quality obtained from moving mesh method is much higher than that from the fixed mesh case, see Fig. 6. As we

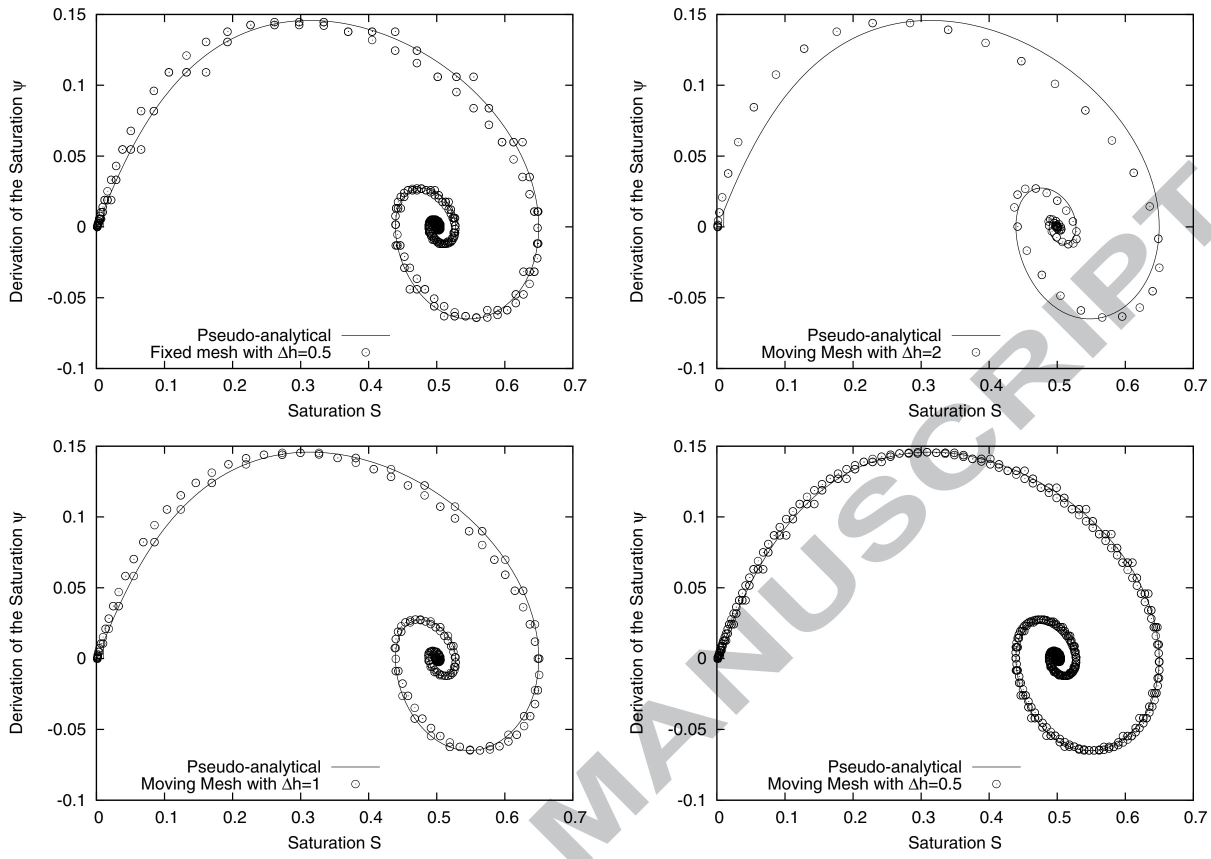

Figure 6: The difference between the pseudo-analytical solution and the numerical solution. The top left one is the numerical solution with a fixed mesh and mesh size of $\Delta h = 0.5$. The top right one, bottom left one and bottom right one are the solutions with the moving mesh technique, the initial mesh sizes are $\Delta h = 2$, $\Delta h = 1$ and $\Delta h = 0.5$ respectively. The parameters are selected the same as in Fig. 5.

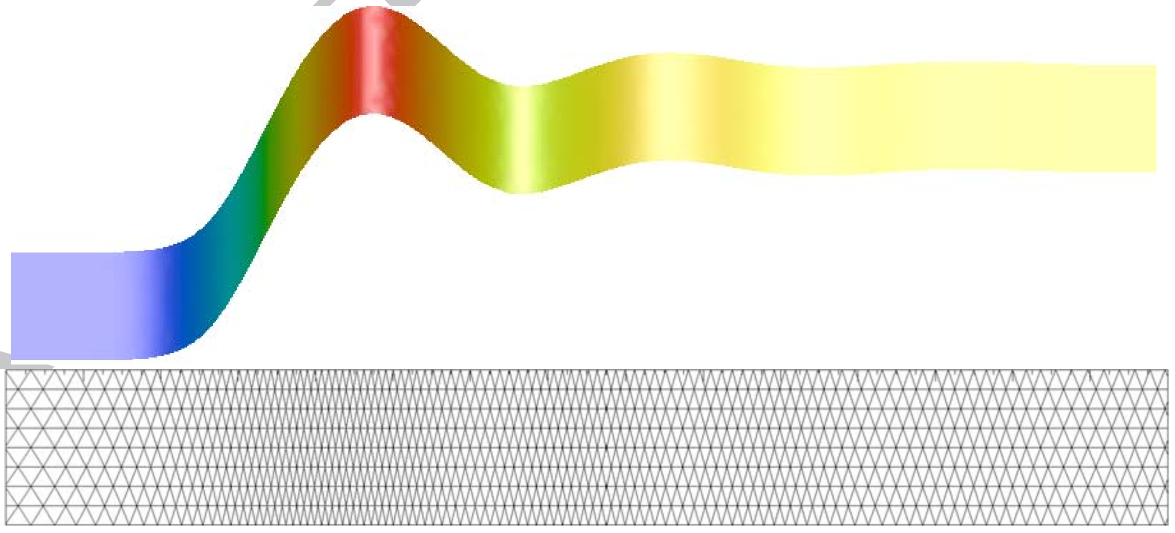

Figure 7: The solution of the NERE model (top) and its mesh (bottom) with the moving mesh technique at $t = 100$. The parameters are selected the same as in Fig. 5.

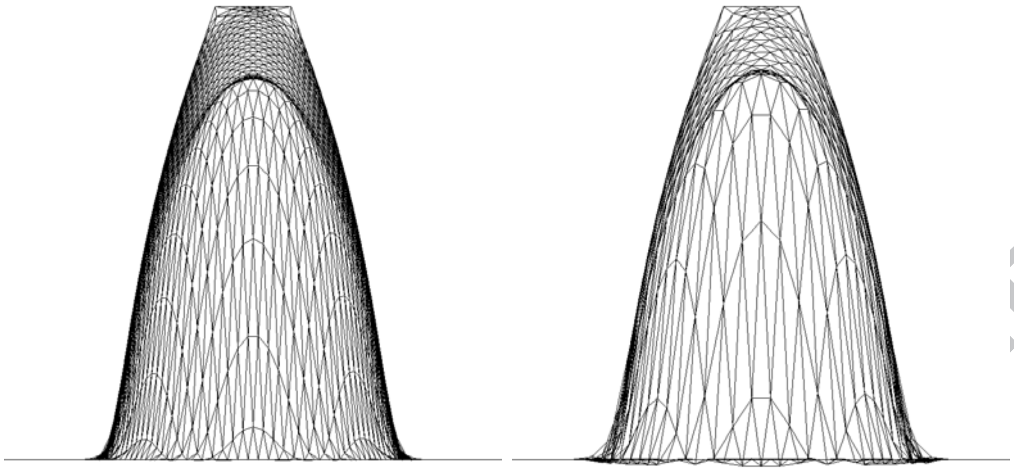
Figure 8: Numerical results are shown at $t = 10$. The parameters: $K(S) = S^2$, $D(S) = S^2$, $\tau = 0.5$ and $\tilde{\delta} = 1$ are used. The left panel shows the view when looking at the finger in the front, where the moving mesh strategy is used. As a comparison, the same view is given on the right hand side with a fixed mesh.

discussed for Fig. 6, the solution quality of the moving mesh method with an initial mesh size $\Delta h = 1.0$ is even better than that of the fixed mesh case with an initial mesh size $\Delta h = 0.5$. The CPU time of the moving mesh method with an initial mesh size $\Delta h = 1.0$ is 56.80 seconds, which is smaller than that of the fixed mesh case with an mesh size $\Delta h = 0.5$. That means that the moving finite element method is more efficient for obtaining the same solution accuracy than with the fixed mesh method.

4.2. Simulating a single finger

In [2], 2D fingers are created through introducing water via a continuous constant point source in a dry soil. In this subsection, we use a similar way to generate fingers numerically.

For the tests in this subsection, the physical domain is $[-5, 25] \times [-10, 20]$. For the first test, the initial condition is taken as

$$S_{init} = 0.15\,(\tanh(2(z-19))\tanh(2(22-z))+1)\,(\tanh(2(x-9))\tanh(2(11-x))+1)+0.1. \tag{36}$$

The physical domain and initial condition are shown in Fig. 2. For the segment $x \in [9, 11]$ on the boundary $z = 20$, the Dirichlet boundary condition $S_{Dir} = 0.7$ is used, and homogeneous Neumann boundary conditions are used for the other boundaries. The initial mesh size is set equal to $\Delta h = 0.5$.

In this test, we use $K(S) = S^2$, $D(S) = S^2$, $\tau = 0.5$, and $\tilde{\delta} = 1.0$. First, we give a comparison between the results obtained with a moving mesh and a fixed mesh. Fig. 8 shows the front views of the finger at $t = 10$. Obviously, spurious oscillations are observed at the bottom of the finger which is obtained with a fixed mesh (Fig. 8, see the right panel). That means that the current mesh size is too rough to resolve the finger successfully. It is impossible to obtain accurate numerical solutions with these oscillations. Of course a global refinement of the mesh may fix this problem, but this will cause an increase of both the CPU time and memory requirement. With the moving mesh method, it is observed from figure (Fig. 8, left one) that those oscillations are removed effectively without changing the amount of mesh grids.

The initial saturation in the dry region is 0.1 in the above simulation. It is pointed out in [7] that a small initial water content leads to compact infiltration fronts, and the size of the overshoot reduces drastically as the initial saturation is increased. Such a phenomenon can also be observed in Fig. 9 where the pseudo-analytical results for 1D case with three different initial saturation are shown. To obtain these results, the parameters $K(S) = S^2$, $D(S) = S^2$, $\tau = 0.5$ are used. From the figure, we can see that with other values of

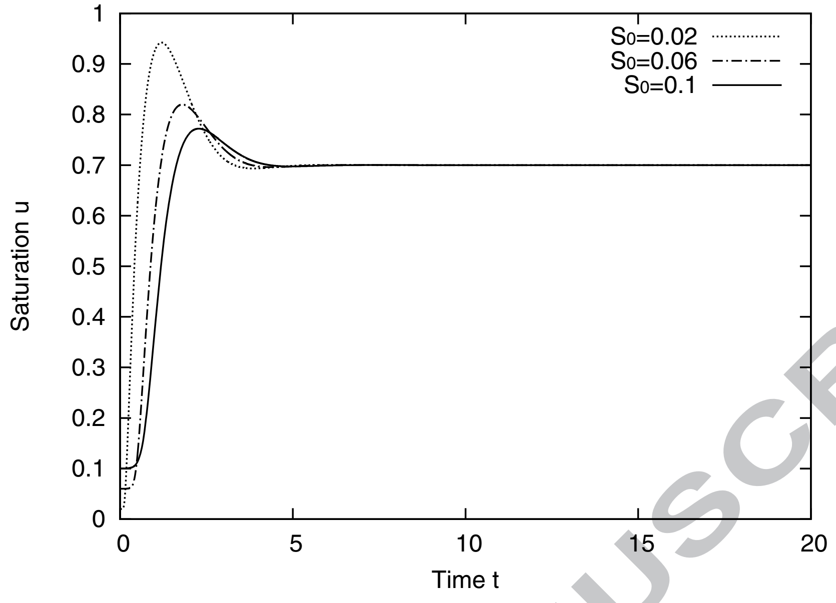

Figure 9: The 1D pseudo-analytical solutions of NERE model with parameters: $K(S) = S^2$, $D(S) = S^2$, $\tau = 0.5$. Three different initial saturation in the dry region are tested.

the parameters, a lower initial saturation results in a higher saturation at the tip of the finger and a sharper wetting front.

In [21], it is concluded that fingers are observed to occur when the initial saturation is below the residual saturation. So an efficient and accurate simulation with numerical experiments with a very small initial saturation is very important.

Fig. 10 shows three results obtained with moving finite element method. In all three simulations, a different initial saturation in the dry region, $S_0 = 0.1, 0.06$, and 0.02, together with the choices $K(S) = S^2$, $D(S) = S^2$, $\tau = 0.5$, $\tilde{\delta} = 1.0$ is used. For $S_0 = 0.1$, the initial function (36) is used. For $S_0 = 0.06$ and $S_0 = 0.02$, the initial functions

$$S_{init} = 0.16\,(\tanh(2(z-19))\tanh(2(22-z))+1)\,(\tanh(2(x-9))\tanh(2(11-x))+1)+0.06 \qquad (37)$$

and

$$S_{init} = 0.17\,(\tanh(2(z-19))\tanh(2(22-z))+1)\,(\tanh(2(x-9))\tanh(2(11-x))+1)+0.02 \qquad (38)$$

are used, respectively. The oversaturation on the tip of the finger, which is the typical phenomenon in the physical experiment, can be observed obviously from Fig. 10 (left column). With a decrease of the initial saturation in the dry region, the wetting front becomes sharper, and the saturation on the tip of the figure becomes bigger, which coincides with the pseudo-analytical solutions in Fig. 9. With the help of the moving mesh method, we can see that grid points are clustered around the finger profiles successfully, and the fingers are resolved without spurious oscillations.

In the three simulations, different initial mesh sizes are being used. For a smaller initial saturation, a much denser mesh is used because of the sharp wetting front. In fact, we can decrease $\tilde{\delta}$ to enhance the adaptivity of the algorithm, then much more grid points will be moved towards the wetting front. Consequently, there is the possibility that a relatively coarse mesh can be used to resolve the sharp fingers. However, a difficulty will arise with the growth of the finger, because the area of the singularity also grows at the same time. So, when the area of the singularity is sufficiently large, or the singularity is quite close to the boundary of the physical domain, there may not be sufficiently enough grid points to resolve the finger any more. According

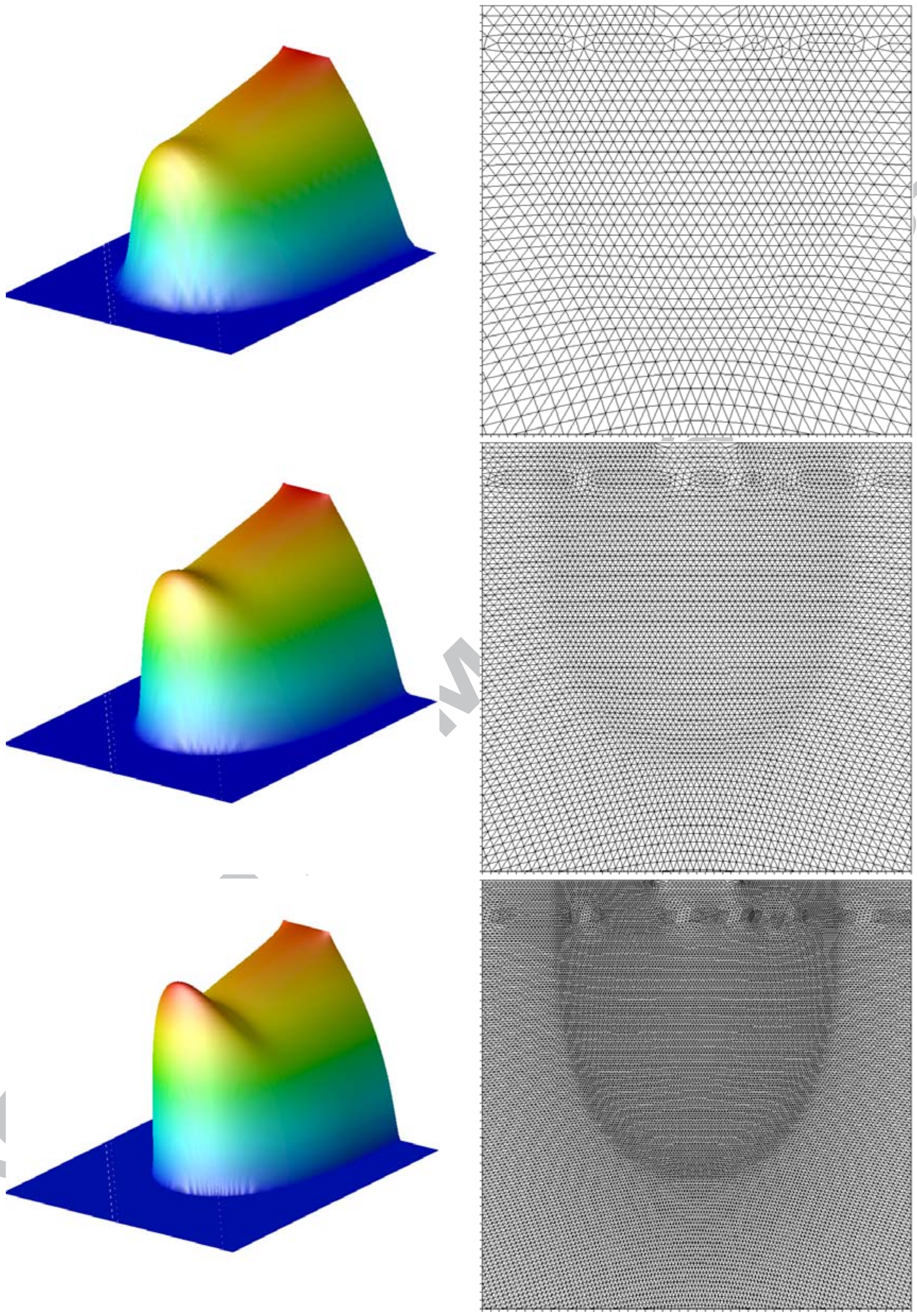

Figure 10: Results obtained with parameters $K(S) = S^2$, $D(S) = S^2$, $\tau = 0.5$, $\tilde{\delta} = 1.0$, and three different initial saturation's in the dry region, $S_0 = 0.1$ (top), $S_0 = 0.06$ (middle), and $S_0 = 0.02$ (bottom). Left: the saturation around the wetting front. Right: the corresponding mesh.

to our numerical experience, a high adaptivity of the mesh may cause an instability of the algorithm. To show a successful simulation of the finger phenomenon with the moving finite element method, a reasonable amount of grid points should be evaluated before hand. Such an evaluation depends on the area of the physical domain, the finger width, etc., which will be investigated in a future paper. We give the references [6, 7, 21] and references therein about how parameters effect the width of the finger.

5. Conclusion

In this paper, the non-equilibrium Richards equation is solved by using a moving finite element method. The standard linear element and second-order Runge-Kutta method are used to discretize the governing equations spatially and temporally. The moving mesh strategy is based on harmonic maps. Compared with the general gradient-based monitor function where the selection of the numerical parameters strongly depends on the underlying problem, a Beckett and Mackenzie type monitor function is used in our paper. A smoothing method which is based on the diffusive mechanism is also used in the algorithm, which improves the mesh quality around the wetting front significantly. With the help of moving mesh methods, a large amount of grid points is clustered around the wetting front. So the hold-back-pile-up phenomenon is detected much better than that when the fixed mesh is used with the same initial mesh size. The numerical convergence of the moving finite element method has been demonstrated in experiments with three successively refined meshes.

In the simulations of the finger phenomenon, the moving finite element method resolved the finger successfully. If the mesh size is not sufficiently small , spurious oscillations are observed when a fixed mesh is used. On the other hand, with the moving mesh method, it is demonstrated that the oscillations are removed or, at least, significantly reduced with the same number of the mesh points. In the simulations, the oversaturation on the tip of the finger, which is a typical phenomenon in the physical experiment, is successfully observed. It is also shown in the numerical results that with a small initial saturation, the saturation on the tip of the finger becomes big, and the wetting front becomes sharp, which coincides with our 1D pseudo-analytical results and with the existing literature.

Compared with the finite element method on a fixed mesh, the moving finite element method is much more powerful for resolving the finger phenomenon. However, based on our numerical experiences, difficulties may arise when the finger is sufficiently large or the wetting front is close to the boundary. Both situations result in the shortage of grid points, resulting in the appearance of spurious oscillations around the wetting front, which cause inaccurate results. To fix these problems, a sufficiently small mesh size should be used according to an investigation of the width of the finger and the area of the computational domain.

Besides r-adaptive methods, there are also h-adaptive and p-adaptive methods which can be used to solve NERE model. Based on our preliminary results obtained with h-adaptive methods, the issue expressed above can be well resolved. With a local refinement technique, the requirement of grid points can always be satisfied even in the region nearby the boundary. While for the region far away from the wetting front, the grid points can be reduced by local coarsening. Combination of r-refinement with h-refinement will be reported in a future paper.

Acknowledgement

We would like to thank Dr. Henk Schuttelaars, Dr. Cas Berentsen and Dr. Ruo Li for the interesting and helpful discussions. The research of Hu is partially supported by a studentship from Hong Kong Baptist University and a China-Utrecht University three-month PhD-Fellowship programme.

References

[1] http://www.gnu.org/software/octave.

[2] T. W. J. Bauters, D. A. DiCarlo, T. S. Steenhuis, and J. Y. Parlange. Soil water content dependent wetting front characteristics in sands. *Journal of Hydrology*, 231-232:244–s54, 2000.

[3] G. Beckett, J. A. Mackenzie, A. Ramage, and D. M. Sloan. Computational solution of two-dimensional unsteady PDEs using moving mesh methods. *Journal of Computational Physics*, 182:478–495, 2002.

[4] W. M. Cao, W. Z. Huang, and R. D. Russell. An r-adaptive finite element method based upon moving mesh pdes. *Journal of Computational Physics*, 149:221–244, 1999.

[5] H. D. Ceniceros and T. Y. Hou. An efficient dynamically adaptive mesh for potentially singular solutions. *Journal of Computational Physics*, 172:609–639, 2001.

[6] L. Cueto-Felgueroso and R. Juanes. Nonlocal interface dynamics and pattern formation in gravity-driven unsaturated flow through porous media. *Physical Review Letters*, 101(244504), 2008.

[7] L. Cueto-Felgueroso and R. Juanes. Stability analysis of a phase-field model of gravity-driven unsaturated flow through porous media. *Physical Review E*, 79(036301), 2009.

[8] A. V. Dam and P. A. Zegeling. Balanced monitoring of flow phenomena in moving mesh methods. *Commun. Comput. Phys.*, 7:138–170, 2010.

[9] G. A. Diment, K. K. Watson, and P. J. Blennerhassett. Stability analysis of water movement in unsaturated porous materials, 1, theoretical considerations. *Water Resour. Res.*, 18:1245–1254, 1982.

[10] A. G. Egorov, R. Z. Dautov, J. L. Nieber, and A. Y. Sheshukov. Stability analysis of traveling wave solution for gravity-driven flow. *Computational Methods in Water Resources, Elsevier Ser. Dev. Water Sci.*, 47:121–128, 2002.

[11] A. G. Egorov, R. Z. Dautov, J. L. Nieber, and A. Y. Sheshukov. Stability analysis of gravity-driven infiltrating flow. *Water Resources Research*, 39:1266, 2003.

[12] G.Beckett and J.A. Mackenzie. Convergence analysis of finite difference approximations on equidistributed grids to a singularly perturbed boundary value problem. *Appl. Numer. Math.*, 35:87, 2000.

[13] G.Beckett and J.A. Mackenzie. On a uniformly accurate finite difference aaproximation of a singularly perturbed reaction-diffusion problem using grid equidistribution. *J. Comput. Appl. Math.*, 131:381, 2001.

[14] R.J. Glass, J.-Y. Parlange, and T.S. Steenhuis. Wetting front instability 1. theoretical discussion and dimensional analysis. *Water Resour. Res.*, 25(6):1187–1194, 1989.

[15] W. G. Gray and S. M. Hassanizadeh. Unsaturated flow theory including interfacial phenomena. *Water Resour. Res.*, 27:1855–1863, 1991.

[16] D. E. Hill and J. Y. Parlange. Wetting front instability in layered soils. *Soil Sci. Soc. Am. J.*, 36:697–702, 1972.

[17] G. H. Hu, Z. H. Qiao, and T. Tang. Moving finite element simulations for reaction-diffusion systems. *Submitted to Journal of Computational Physics.*

[18] R. Li and W. Liu. http://dsec.pku.edu.cn/∼rli.

[19] R. Li, T. Tang, and P. W. Zhang. Moving mesh methods in multiple dimensions based on harmonic maps. *Journal of Computational Physics*, 170:562–588, 2001.

[20] R. Li, T. Tang, and P. W. Zhang. A moving mesh finite element algorithm for singular problems in two and three space dimensions. *Journal of Computational Physics*, 177:365–393, 2002.

[21] J. L. Nieber, R. Z. Dautov, A. G. Egorov, and A. Y. Sheshukov. Dynamic capillary pressure mechanism for instability in gravity-driven flows; Review and extension to very dry conditions. *Transport in Porous Media*, 58:147–172, 2005.

[22] F. Otto. l^1-contraction and uniqueness for unstationary saturated-unsaturated water flow in porous media. *Adv. Math. Sci. Appl.*, 7:537–553, 1997.

[23] Z. Qiao. Numerical investigations of the dynamical behaviors and instabilities for the Gierer-Meinhardt system. *Commun. Comput. Phys.*, 3:406–426, 2008.

[24] L. A. Richards. Capillary conduction of liquids through porous mediums. *Physics*, 1(5):318–333, 1931.

[25] G. C. Sander, O. J. Glidewell, and J. Norbury. Dynamic capillary pressure, hysteresis and gravity-driven fingering in porous media. *Journal of Physics: Conference Series*, 138, 2008.

[26] J. H. Smith and A. M. Stuart. Analysis of continuous moving mesh equations. *Technical Report, SCCM Program*, 1996. Stanford University, Stanford, CA.

[27] H. Z. Tang and T. Tang. Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 41:487–515, 2003.

[28] T. Tang. Moving mesh methods for computational fluid dynamics. *Contemporary Mathematics*, 383, 2005.

[29] H. Y. Wang and R. Li. Mesh sensitivity for numerical solutions of phase-field equations using r-adaptive finite element methods. *Communications in Computational Physics*, 3(2):357–375, 2008.

[30] H. Y. Wang, R. Li, and T. Tang. Efficient computation of dendritic growth with r-adaptive finite element methods. *Journal of Computational Physics*, 227(12):5984–6000, 2008.

[31] P. A. Zegeling. On resistive MHD models with adaptive moving meshes. *Journal of Scientific Computing*, 24:263–284, 2005.

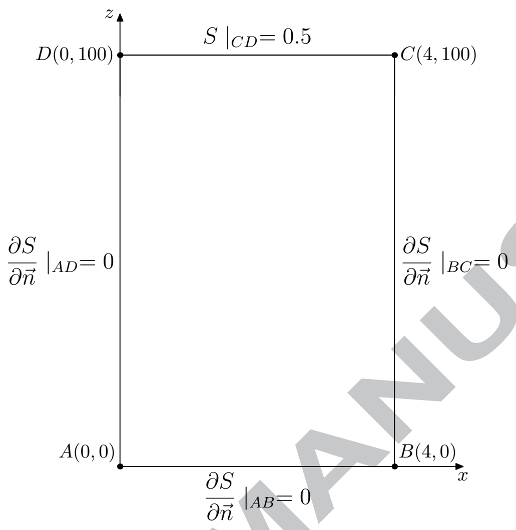
z
$S\mid_{CD}=0.5$
$D(0,100)$
$C(4,100)$
$\frac{\partial S}{\partial \vec{n}}\mid_{AD}=0$
$\frac{\partial S}{\partial \vec{n}}\mid_{BC}=0$
$A(0,0)$
$B(4,0)$
x
$\frac{\partial S}{\partial \vec{n}}\mid_{AB}=0$

Figure 4: The computational region