A model based on cellular automata for the parallel simulation of 3D unsaturated flow

Gianluigi Folino [a], Giuseppe Mendicino [b], Alfonso Senatore [b], Giandomenico Spezzano [a,*], Salvatore Straface [b]

[a] *Institute for High-Performance Computing and Networking (ICAR) - CNR, P.te Pietro Bucci 41C, I-87036 Rende (CS), Italy*

[b] *Department of Soil Conservation, University of Calabria - P.te Pietro Bucci 41B, I-87036 Rende (CS), Italy*

Abstract

Cellular automata (CA) are discrete dynamic systems that have been used for modeling many physical systems. CA are often used as an alternative to model and solve large-scale systems where the use of partial differential equations involve complex and computationally expensive simulations. The purpose of this work is to investigate the use of CA based techniques for modeling and parallel simulation of water flux in unsaturated soils. Unsaturated flow processes are an important topic in several branches of hydrology, soil science and agricultural engineering dealing with soil-atmosphere interaction, subsurface flow and transport processes. In this paper a CA model for 3D unsaturated flow simulation is proposed using an extension of the original computational paradigm of cellular automata. This model, aimed at simulating large-scale systems, uses a macroscopic CA approach where local laws with a clear physical meaning govern interactions among automata and its correctness is proved by CAMELot system, which allows the specification, parallel simulation, visualization, steering and analysis of CA models in the same environment, using a friendly interface providing at the same time considerable flexibility. The model has been validated with reference multidimensional solutions taken from benchmarks in literature, showing a good agreement even in the cases where non-linearity is very marked. Furthermore, using some of these benchmarks we present a scalability analysis of the model and different quantization techniques aimed at reducing the number of messages exchanged, and then the execution time, when simulations are characterized by scarce mass interactions.

Key words: unsaturated flow modeling, cellular automata, parallel simulation environments, performance model, quantization techniques.

1 Introduction

Computer models are widely used in many disciplines, ranging from physics, chemistry, biology, economics, to hydrology. A computer model is a computer program which attempts to simulate an abstract model of a particular system. Many researchers use computer modelling as the principal tool for qualitative and quantitative insight into complex systems and as a virtual laboratory to explore theory by simulation. Today, we can build simulation models which allow to predict the behavior of physical systems in actual or hypothetical circumstances. Moreover, the advent of parallel computing is making complex simulations in many disciplines tractable, increasing the interest in different application fields.

An area of remarkable interest where the importance of simulation is increasing is the one concerning fluid flow in unsaturated zones. Unsaturated flow processes are an important topic in several branches of hydrology, soil science and agricultural engineering dealing with subsurface flow and transport processes. These phenomena are in general complicated and difficult to describe quantitatively, since they often entail changes in the state and content of soil water during flow.

Soil water dynamics can be expressed in the form of mathematical expressions aimed at describing the hydrological relationships within the system. The governing equations define a mathematical model. The entire model has usually the form of a set of partial differential equations, together with auxiliary conditions. The auxiliary conditions have to describe the system geometry, the system parameters, the boundary conditions and, in case of transient flow, also the initial conditions. If the governing equations and auxiliary conditions are simple, an exact analytical solution may be found. Otherwise, a numerical approximation is applicable. The numerical simulation models are by far the most applied ones.

Numerical simulations require the discretization of the model into a grid. This approach is very sensitive to temporal and spatial resolution and has serious drawbacks. For example, using explicit finite difference methods in order to obtain reasonable accuracy, the length of the interval in space must be kept small. Furthermore, to get a stable solution, the time step has to be small compared with the space interval. Thus, it is necessary to have a large number of time steps when the simple explicit method is used.

An alternative approach is to consider a discrete cell system characterized by smaller sizes of cells where the uniformity hypothesis is less respected,

* Corresponding author.
Email address: spezzano@icar.cnr.it (Giandomenico Spezzano).

then we can use the same constitutive equations adopted in the differential context as an approximation, without the need of going down to the differential form and then going up again to the discrete form [18]. Such an approach is particularly suitable to the use of cellular automata (CA) paradigm [21] and to be developed in parallel computing environments.

CA are discrete dynamic systems composed of a set of cells in a regular spatial lattice, either one-dimensional or multidimensional. They are characterized by the following properties: each cell can have any one of a finite number of states; the states of all cells are updated synchronously according to a local rule, called transition function; the evolution takes place in discrete time steps. The collection of states of all the cells of the lattice forms the *configuration* of the automaton.

CA model a problem in a naturally parallel way as a collection of identical transition functions (set of rules) simultaneously applied to all the cells of the automata. This bottom-up approach differs from standard one to parallel computation, where a problem is split into independent subproblems, each solved by a different processor, and then combined in order to yield the final solution.

Many CA applications in fluid-dynamics exist, most of these based on *microscopic* approaches: lattice gas automata models were introduced to describe the motion and collision of *particles* interacting according to appropriate laws on a lattice space [5]. Lattice gas models, due to the simplicity of both fluid particles and their interactions, allow simulations of a large number of particles, which are only capable of reproducing the birth of macroscopic flow patterns. The discrete nature of lattice gas models has shown some weaknesses which can be partially solved by the Lattice Boltzmann models, where the state variables can take continuous values (instead of integer variables), as they are supposed to represent the density of fluid particles endowed with certain properties located in each cell [17]. Lattice Boltzmann models have been used to model flow in porous media, but the applications so far developed take a more microscopic approach and aim at describing phenomena which take place at the pore level [1].

Unsaturated flow modelling represent complex macroscopic fluid dynamic phenomena, which seem difficult to model in these CA frames, because they occur on a large space scale and need, practically, a macroscopic level of description that involves the management of a large amount of data. Empirical CA methods were developed on the macroscopic scale in order to overcome these limits [7], using local laws where automata interactions were based on parameters whose physical meaning was not clear and, as a consequence, heavy calibration phases were necessary to estimate suitable values of the same parameters.

In this paper by means of the same extended notion for *macroscopic* cellular automata, we describe a three-dimensional model which simulates water flux in unsaturated not deformable soils, but the local laws governing the automata interactions are based on physically-based rules.

Although CA provide an intrinsically parallel model, to use this model effectively on a parallel machine a programming environment for macroscopic approaches has to be considered. To this purpose, a problem solving environment called CAMELot that allows interactive simulation and steering of parallel cellular computations has been utilized [3]. CAMELot is a system that uses macroscopic CA model both as a tool to model and simulate dynamic complex phenomena and as a computational model for parallel processing. It combines simulation, visualization, control and parallel processing into a tool that allows to interactively explore a simulation, visualize the state of the computation as it progresses and change parameters, resolution or representation on the fly.

The 3D unsaturated flow model built in CAMELot is based on the application in each cell, which represents a portion of the soil, of a partially discrete form of the physical laws obtained by combining a mass-balance equation and a constitutive equation (Darcy's law). This approach allows both to provide a physically-based method for modeling the phenomenon and to maintain the transition function rules simple. The accuracy of the model has been evaluated for different multidimensional schemes providing results similar to those of other approaches described in Paniconi et al. [14] and Smith et al. [15].

Furthermore, for the same model an analysis of the performance was carried out on a 3D scheme characterized by different sizes. Numerical tests have shown that when the CA unsaturated flow model is used to simulate larger domains, or by analyzing greater spatial resolution, then the execution performance has a good scalable behavior. All the same, the execution times of the model have been reduced using different quantization techniques [22] aimed at limiting the number of messages exchanged, especially when scarce mass interactions occur. A quantized system is a system with input and output quantizers, which generates state updates only at quantum level crossings. Unfortunately, the quantization techniques introduce errors because of the approximations adopted. Combining different methods, with regard to time and space, we have obtained important reductions in the error involved, while maintaining the high performance of quantized models.

Section 2 presents a discrete approach describing fluid flow in unsaturated soils, obtained combining a mass-balance equation with a constitutive equation. Section 3 describes the mapping of the discretization in a macroscopic cellular automata scheme. Section 4 presents the CAMELot environment and an analytical performance model. Section 5 shows the simulations carried out

within CAMELot, the model accuracy and the validation of the performance. Finally, simulations of the quantized versions of the model are shown in Section 6.

2 A discrete formulation for flow modelling in unsaturated soils

If a discrete cell system is considered, we may hypothesize the form of the governing equation of the unsaturated flow through the same constitutive law used in the differential context, as an approximation. This equation, whose solution provides the variables configuring the infiltration phenomenon, is based on the interpretation of the mass-balance equation and the constitutive equation (Darcy's law), and it can be written in the following complete form according to Mendicino et al. [12]:

$$\sum_{\alpha} -K_{\alpha c}(\psi_c)\left(\frac{h_\alpha - h_c}{l_\alpha}\right)A_\alpha + V_c C_c \frac{\Delta h_c}{\Delta t} = q_c \tag{1}$$

where pedices $_c$ indicates that the variable is referred to the cell c where the water mass balance is performed, while pedices $_\alpha$ indicates that the variable is referred to the adjacent cell along the generic direction α obtained linking the centers of the masses of the two cells, $K_{\alpha c}(\psi_c)$ [LT^{-1}] is the hydraulic conductivity averaged between cell c and the adjacent cell along the direction α (it is constant in a saturated soil while in an unsaturated soil it depends on capillary pressure ψ_c [L]), h_c and h_α [L] are the total heads of the cell c and of that adjacent, l_α [L] is the cell dimension, A_α [L^2] is the surface area where the flux passes through, V_c [L^3] is the cell volume, C_c [L^{-1}] is the specific retention capacity depending on ψ_c [2], $\frac{\Delta h_c}{\Delta t}$ is the total head gradient in the time step Δt, and q_c is the volumetric mass source term [$\mathrm{L}^3\mathrm{T}^{-1}$].

Equation 1 can be applied to each cell of the domain in the same way as explicit-type approaches, where Δh_c is the only unknown term to be estimated in the next time step. Such an approach proves to be particularly suitable to the use of a macroscopic cellular automata environment (such as will be described in the next section) and to be developed in a parallel computing system.

The Delaunay tessellation used in our case involves a discrete governing equation similar to the one achieved using Finite Difference or Finite Volume Method schemes [11] [10]. If we don't use a Delaunay tessellation and an irregular mesh is considered, then the discrete governing equation is achieved by means of an interpolation of the hydraulic head on the cells of the automata. In this case Tonti [18] has shown that for linear interpolation the discrete gov-

erning equation system coincides with the one of the Finite Element Method, but for quadratic interpolation it differs from the FEM scheme, also achieving a greater convergence order.

Furthermore, when a method with a differential or integral formulation is used, the choice of the cell or the tessellation type depends on the method selected. Differently, in the case of direct discrete formulation cell, tessellation and time interval can be chosen considering the physical laws governing the problem, the spatial and temporal scale of the phenomenon and, finally, the macroscopic CA environment (for example, by using different shapes - triangles, squares and hexagons - for the single elements in the same domain).

The discrete formulation that we present can be applied to heterogeneous and anisotropic porous media while avoiding jump conditions. The hydraulic head is assumed to be continuous and need not be differentiable. Each cell may have different constitutive properties: this allows composite porous or fractured media to be dealt with. The sources can be discontinuous (drainage trench or superficial recharge) and also may be concentrated (wells or sinks). Equation 1 is valid for interior or boundary cells: in this way, we can avoid the unnatural separation of the differential equations and the boundary conditions, which is typical of a differential formulation.

3 The cellular automata model for 3D unsaturated flow

Cellular automata (CA) are a dynamic system where space, time and states are hypothesized as being discrete. They are based on a division of space in regular cells, each one having an identical computational device embedded: the finite automaton (*fa*). The *fa* input is given by the states of the neighboring cells, including the cell embedding the *fa*. Specifically, the *fa* states vary according to a local rule (transition function); i.e. in a given time step a *fa* state depends on its state and on those of neighboring cells at the previous time step. Finite automata have identical transition functions, which are simultaneously applied to each cell (synchronous CA). At the beginning all the *fa* are in arbitrary states representing the initial conditions of the system, then the CA evolves by changing the state of all *fa* simultaneously at discrete time steps according to the *fa* transition function. Finally, the global evolution of the CA system is derived from the evolution of all the cells.

There are several modifications and extensions of the standard CA which have been considered in literature **??**. Generalized CA models include some of the following features:

- temporal and spatial heterogeneity both in the transition function and in the

neighborhood. Temporal heterogeneity is used when the transition function or the neighborhood of a cell changes during time. An example for temporal heterogeneity occurs in picture processing by CA where the pixels of a digitized image are stored in the cells of a CA, and the processing consists in the application of several filters or other operators to an image, where each of the operators is implemented as the local transition rule of a CA. In this case it helpful to use a generalized CA where first for some number of steps the local rule implementing the first operator is applied, afterwards to the result of this, i.e., the contents of the cells of the CA, for some more steps the local rule implementing the second operator, and so on. Spatial heterogeneity occurs when one can allow that the size and the shape of the neighborhood of a cell may vary throughout the lattice, and similarly the local transition rule used by the cells.

- continuity of the state. Continuous CA are an extension of CA in which the discrete state values of CA cells are replaced with continuous real values. Like basic CA, continuous CA are discrete in space and time and are a versatile technique for modeling a wide variety of phenomena. The continuity of the state is not a problem; in practical case the utilized variables have a finite number of significant digits and a finite range of permitted values, then the set of utilized values could be extremely large, but always finite.
- asynchrony in the transition function so that every cell can, at each step, nondeterministically choose between changing its state according to σ or keeping it;
- complex time-dependent neighborhoods (i.e. block rules), and
- probabilistic and hierarchical transition functions.

The proposed model consists of a continuous CA characterized by a three-dimensional domain, regularly subdivided in cubic cells described by the following functional structure:

$$\mathbf{A} = (E^d, X, Q, P, \sigma) \tag{2}$$

where $E^d = \{(x, y, z) | \, x, y, z \in N, 0 \leq x \leq l_x, 0 \leq y \leq l_y, 0 \leq z \leq l_z\}$ is the set of cells identified by points with integer co-ordinates in the finite region, where the phenomenon evolves; N is the set of natural numbers; l_x, l_y and l_z represent the limits of the region;

$X = \{(0,0,0), (-1,0,0), (0,1,0), (0,0,-1), (0,0,1), (0,-1,0), (1,0,0)\}$ identifies the 3D von Neumann neighborhood, which influences the change in state of the central cell (Fig. 1);

Q is the finite set of the *fa* states, given by the Cartesian product of the following sub-states:

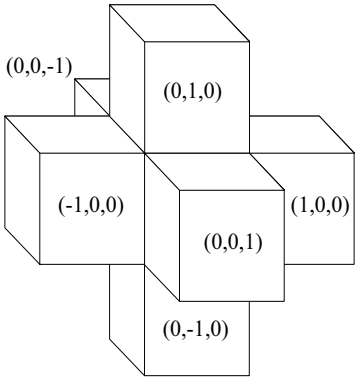

Fig. 1. Three-dimensional von Neumann neighborhood.

$$Q = Q_h \times Q_\psi \times Q_\theta \times Q_K \times Q_\partial \tag{3}$$

where Q_h is the sub-state describing the total head of the cell, Q_ψ is the sub-state describing the pressure head ($\psi = h - z$), Q_θ is the sub-state describing the water content through the value of moisture content in volume θ, Q_K describes the hydraulic conductivity sub-state and, for transient condition analyses Q_∂ indicates the sub-state corresponding to a parameter value necessary to guarantee the convergence of the system;

P is the finite set of CA global parameters which affects the transition function and is made up of some parameters associated to the characteristic equations, the saturation permeability, the automaton dimension and the time step. In particular, we have the residual water content θ_r, the saturation water content θ_s, the capillary air entry pressure ψ_s, the pore-dimension distribution index n, a continuity parameter for pressure head ψ_0, the specific storage S_s, the saturated hydraulic conductivity K_s, the cell dimension l_α and the time step Δt;

$\sigma : Q^7 \rightarrow Q$ is the deterministic transition function. Once the initial conditions (total head, pressure head, conductivity and water content values) and the boundary conditions are fixed, it is based on two elementary steps:

- σ_1: the update of the soil hydraulic characteristics (i.e. the hydraulic conductivity K, the water content θ_c and the specific retention capacity C_c), depending on the pressure head through the characteristic equations;
- σ_2: the application of the unsaturated soil flux equation 1, to update the values of the total head h_c and the pressure head ψ_c of the cell.

Starting the simulation, the sub-states condition depends on the initial values

assigned to the total head h_c, while the boundary conditions can be assigned either in terms of mass flow coming in (infiltration) or out (exfiltration) from the system (Neumann conditions), or fixing the total or pressure head values on some cells of the system (Dirichlet conditions).

Once the initial and boundary conditions of the problem are defined, and the global parameters are initialized, then the transition function can be applied to the cells of the system. Further on the two elementary steps allowing the evolution of the system in time (schematized in figure 2) will be analyzed.

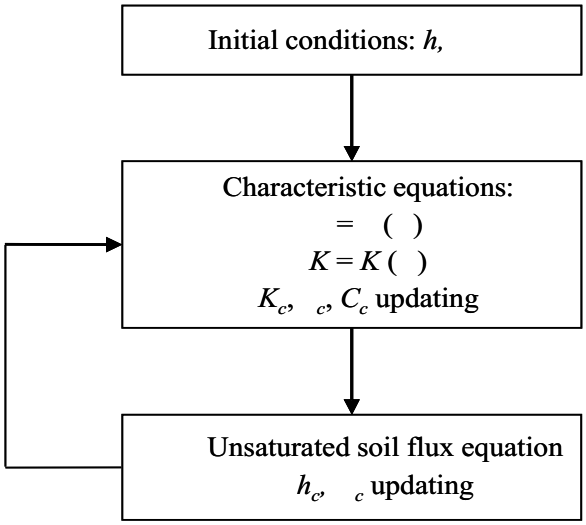

Fig. 2. Schematization of the two elementary steps allowing the evolution of the CA.

3.1 Update of the soil hydraulic characteristics (σ_1)

In equation 1 it is necessary to specify the non-linear dependencies among the assumed independent variable, total head h_c, and terms characterizing the hydraulic properties of the soil represented by the hydraulic conductivity $K(\psi_c)$, the water content θ_c and the specific retention capacity C_c. The hydraulic properties of the soil are essentially expressed by retention curves, which in the case of moisture content are in a form like $\theta = \theta(\psi)$, while for relative hydraulic conductivity the relationship $K = K(\psi)$ is obtained from physically-based models. Many theoretical models for the constitutive equations $\theta = \theta(\psi)$ and $K = K(\psi)$ are available in literature.

The update of the hydraulic properties of the soil is not only necessary to allow the evolution of the system, but also for the determination of the convergence conditions. Infact, it can be easily proved that in a three-dimensional scheme, for $\Delta x, \Delta y, \Delta z \to 0$, with the permeability along the three orthogonal directions around the cell c varying linearly, the solution of the equation

1 converges when t is finite and $\Delta t \leq \frac{C(\psi) l_\alpha^2}{6K(\psi)}$, showing that Δt value depends on $C(\psi)$ and $K(\psi)$.

3.2 Application of the unsaturated soil flux equation (σ_2)

If the hydraulic properties of the soil are updated, then the unsaturated soil flux equation 1 can be applied to obtain for the analyzed cell the new value of the independent variable, the total head h_c, from which all sub-states are depending on. The unknown term in the equation 1 is Δh_c, indicating the total head variation in the cell between instants t and $t + \Delta t$. This term is achieved by the interaction of the analyzed cell with the neighboring ones. Considering the generic α direction, the same total head variation is given by the following equation:

$$\Delta h_{\alpha c} = \frac{\Delta t}{V_c C_c} \left[q_c + K_{\alpha c}(\psi_c) \left(\frac{h_\alpha - h_c}{l_\alpha} \right) A_\alpha \right] \tag{4}$$

Then, the total head variation Δh_c due to all neighboring interactions is obtained as follows:

$$\Delta h_c = \sum_\alpha \Delta h_{\alpha c} \tag{5}$$

Hence, the new value of the total head of the cell in the instant $t + \Delta t$ results:

$$h_c \equiv h_c + \Delta h_c = h_c + \sum_\alpha \Delta h_{\alpha c} \tag{6}$$

From equation 6 the pressure head ($\psi = h - z$) can be determined to update again the soil hydraulic properties and to apply the equation 1 for the next time step.

An example of application of the equation 1 is schematized in figure 3 for a simple bi-dimensional case, where for the sake of simplicity source terms are neglected and terms depending on spatial dimensions are simplified considering the cubic structure of the CA.

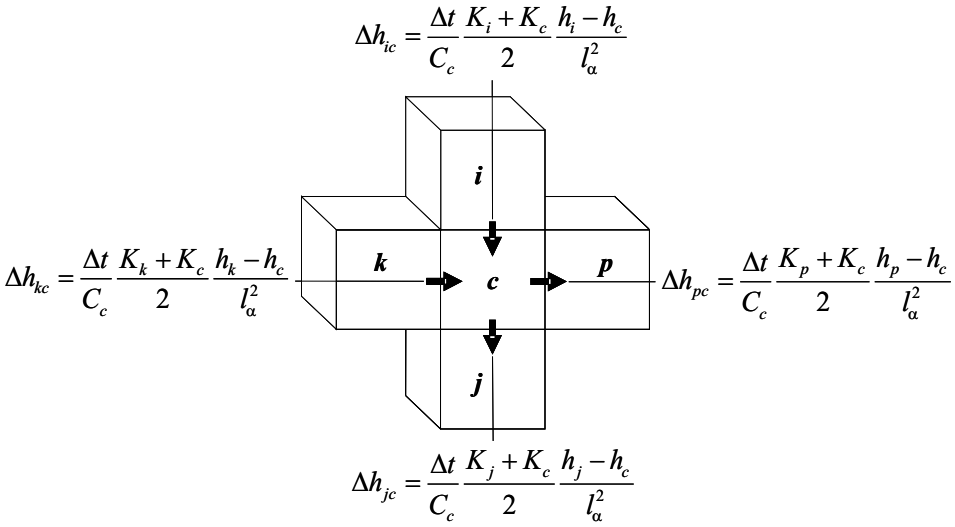

Fig. 3. Transition function applied to the cell c in a bi-dimensional infiltration case, where $h_i, h_k > h_c$ and $h_j, h_p < h_c$.

4 CAMELOT: a software to simulate CA models

CAMELOT is a high performance simulation environment to model complex systems based on the CA formalism. CAMELot allows to accurately model many real-world problem by generalized CA. In our approach, a cellular algorithm is composed of all the transition functions of the cells that compose the lattice. Each transition function generally uses the same local rule, but it is possible to define some cells with different transition functions. Unlike early cellular approaches, in which cell state is defined as a single bit or a set of bits, CAMELot defines the state of a cell as a set of typed sub-states. This allows extending the range of applications to be programmed by cellular algorithms. Furthermore, we introduce a logic neighborhood that may represent a wide range of different neighborhoods inside the same radius and that may also be time-dependent. We have also implemented some mechanisms to observe and control the evolution of the automaton. The run-time support is implemented as a SPMD (Single Program, Multiple Data) program. The latest implementation is based on the C language plus the standard MPI library and can be executed on different parallel machines and cluster of workstations using the Linux operating system. A copy of CAMELOT environment can be downloaded at http://www.icar.cnr.it/spezzano/camelot/camelot.html.

The CAMELot simulation environment consists of:

- a graphic user interface (GUI) for editing, compiling, configuring, executing, visualizing and steering the computation. The GUI allows, by menu pops, to define the size of the CA, the number of the processors on which the automaton must be executed, and to choose the colors to be assigned to the

cell sub-states to support the graphical visualization of their values;
- a software library to integrate raster GIS images into the CA. The raster information can consist of different variables such as altimetry, soil, temperature, vegetation, etc. In CAMELot these variables are associated with the sub-states where the transition function provides a dynamic alteration of the information;
- a load balancing algorithm similar to the scatter decomposition technique to evenly distribute the computation among processors of the parallel machine;
- a language, called CARPET [16], which can be used to define cellular algorithms and to perform steering commands when complex space and time events are detected.

CARPET is a language to program cellular algorithms and contains constructs to extend the range of interaction among the cells, introducing the concept of region, and to define algorithms to perform computational steering. It is a high-level language based on C with additional constructs to describe the rule of the state transition function of a single cell of a cellular automaton and steer the application. In CARPET the state of a cell is described as a record of typed sub-states (*char*, *shorts* (16 bits integers), integers, floats (reals), doubles (64 bits real) and mono-dimensional *arrays*. CARPET program is composed of a *declaration* part that appears only once in the program and must precede any statement, a *body* program that implements the transition function, and a *steering* part that contains a set of commands to extract and analyze system information and to perform steering. Figure 4 shows an example of application of these constructs. Two 3D regions are defined in a three-dimensional cellular automata. The event expression checks whether the maximum and the minimum of the rainfall sub-state in a region (zone1) are equal. If they are equal, the computation is stopped. If the sum of the rainfall values in another region (zone2) is greater than a threshold, then the alpha parameter value is changed. In any case, the computation is stopped after 10000 generations.

4.1 Performance Analysis of the model

In this section a performance model used to predict the scalability and the execution time of a CA model simulated by CAMELot is described. A validation of the performance model is presented in the experimental section. Analyzing the performance of a given parallel algorithm/architecture requires a method to evaluate the scalability. The *isoefficiency* function [6] is one among many parallel performance metrics that measure scalability. It indicates how the problem size n must grow as the number of processor m increases in order to obtain a given efficiency E. It relates problem size to the number of processors required to maintain the efficiency of a system, and lets us to de-

```
{
 ....
  state (float rainfall, infiltration);
  parameter (alpha 2.0, ....);
  .....
  region (zone1(10:20, 1:30, 5:50), zone2(1:4, 3:10, 1:50));
 }
  .....
steering
{
  if (step > 10000)
      cpt_abort();
     if (region_max(zone1, rainfall) == region_min(zone1,rainfall)
       cpt_abort();
     else if (region_sum(zone2, rainfall) > Threshold)
       cpt_set_param(alpha, 3.5);
     .....
 }
```

Fig. 4. An example of use of steering commands in CARPET.

termine scalability with respect to the number of processors, their speed, and the communication bandwidth of the interconnection network. Here, we use the isoefficiency function to analyze the scalability of CA models simulated by the CAMELot environment.

We assume that a CA model is represented in CAMELot by a grid of size $A \times B \times C$ cells, where A is the width, B is the height and C is the depth of the grid. Furthermore, d represents the size (in bytes) of the state of a cell. On a sequential machine we can model the computation time T_1 for one time-step of the cellular automata as:

$$T_1 = t_{as} + ABC(t_f + t_{up})$$

where t_f is the average computation time required to perform the transition function on a single grid point; t_{as} is the average time required at each step to perform some simple operations, such as the increment of the iterations and the zero setting of some variables; t_{up} is the average time necessary to update the state of each cell of the grid with the new values. So, defining $t'_f = t_f + t_{up}$ we have:

$$T_1 = t_{as} + ABCt'_f \tag{7}$$

In the parallel implementation, the grid of the CA is decomposed along the horizontal dimension, and partitioned among p tasks, with each task responsible for a subgrid of size $\frac{A}{p} \times B \times C$. Each processor has allocated a task performing the same computation on each point and at each time step. Each

subgrid is bordered in order to permit a local selection of the neighbors. The main operations of the parallel algorithm for the run-time execution of the CA are summarized in figure 5.

```
foreach time-step
        foreach processor
                foreach z,y,x
                        transition_function(x,y,z,step)
                end foreach
                copy_borders
                exchange_borders
                copy_CA
        end foreach
end foreach
```

Fig. 5. The parallel algorithm for the run-time execution of the CA.

The parallel execution time of the parallel CA on a parallel machine with p processors can be modelled by summing up the computation time of all these functions as follows:

$$T_p = (t_{as} + \frac{A}{p} BCt'_f) + T_{cb} + T_{eb} \tag{8}$$

where T_{cb} is the time spent to copy the borders into a linearized data structure and T_{eb} is the time required to exchange the borders. In fact, each task, before the execution of an iteration must send the borders of own portion of the automata to the two neighboring tasks and receive the correspondent one from the same tasks for a total of four messages and 4 $B \times C \times d$ data. The time required to exchange the borders, according to the Hockney's model [8], is:

$$T_{eb} = 4(t_s + BCdt_b)$$

where t_s is the startup time, that is, the time required to initiate the communication, and t_b is the incremental transmission time per byte, which is determined by the physical bandwidth of the communication channel linking the source and destination processors.

The time spent by the copy_border function depends on the dimension of the portion of the automata lying on a processor and on the total number of substates, so it can be represented as an additive constant plus a linear function of these variables.

Therefore we can represent the parallel execution time as:

$$T_p = t_{ap} + \frac{A}{p} BCt''_f + 4BCdt_b \tag{9}$$

where t_{ap} includes all the times that do not depend on the size of the grid and $t_f^{''} = t_f^{'} + dt_{cb}$; dt_{cb} is the time for copying a cell of data (d is the number of bytes and t_{cb} is the time necessary to copy a byte).

The speedup on p processors can be evaluated as $S = \frac{T_1}{T_p}$ and the efficiency as $E = \frac{S}{p}$. Using the expressions 7 and 9, the speedup and the efficiency can be expressed respectively as:

$$S = \frac{T_1}{T_p} = \frac{t_{as} + ABCt_f'}{t_{ap} + \frac{A}{p}BCt_f^{''} + 4BCdt_b} \tag{10}$$

$$E = \frac{S}{p} = \frac{t_{as} + ABCt_f'}{ABCt_f^{''} + p[4BCdt_b + t_{ap}]} \tag{11}$$

The overhead function T_0 of a parallel system represents the total sum of all overhead incurred by the p processors during the parallel execution of the algorithm and it depends on the problem size. If some really plausible hypothesis are verified, i.e. t_{as} is negligible with respect to the total time and that $t_f^{''} \cong t_f^{'}$, then T_0 will be given by:

$$T_0 = pT_p - T_1 \cong p(t_{ap} + 4BCdt_b) \tag{12}$$

For scalable parallel systems, the problem size T_1 must grow to keep the efficiency fixed, as p increases. To maintain efficiency to fixed value (between 0 and 1), the following relation must be satisfied:

$$T_1 = kT_O \tag{13}$$

where $k = \frac{E}{1-E}$ is a constant depending on the efficiency to be maintained. From expression 7 and 12 we obtain:

$$ABCt_f' \cong kp(t_{ap} + 4BCdt_b) \tag{14}$$

The isoefficiency function is determined by abstracting the problem size as a function of p, through algebraic manipulations in the equation 13. In our case, examining the equation 14, we can notice that doubling the number of processors, we must double the width of the grid, in order to maintain the efficiency to a constant value. Since the amount of computation increases linearly with respect to p in order to keep the efficiency constant, our implementation is highly scalable. From equation 10 we deduce that for a fixed problem size, the speedup saturates at:

$$\frac{t_{as} + ABCt_f'}{t_{ap} + 4BCdt_b} \tag{15}$$

when increasing to infinity the number p of processors.

5 Numerical tests

Tests have been conducted in order to demonstrate that the cellular automata based model achieves solutions very close to those obtained using the classical techniques. Specifically, two benchmarks (one- and two-dimensional cases) were used. Furthermore, the performance model, described in the previous section, is validated on a three-dimensional benchmark, obtained by considering the two-dimensional test area as a section of a three-dimensional plot, with non-homogeneous boundary conditions. Finally a scalability analysis is performed.

5.1 Evaluating the accuracy of the benchmarks

The one-dimensional benchmark used to evaluate the accuracy of the model is described in [14]. It refers to an infiltration problem along a soil column and it is based on the original Richards' equation which does not allow a closed analytical solution.

In order to verify the goodness of the model and for the estimate of errors produced by the CA model some reference numerical solutions were considered, assuming a very dense grid and small time step.

The characteristic equations $\theta = \theta(\psi)$ and $K = K(\psi)$ taken into account for these simulations are those proposed by van Genuchten and Nielsen [20], with the retention curve $\theta(\psi)$ modified as suggested by Paniconi et al. [14].

The accuracy of the CA model was estimated by using both a first:

$$\varepsilon 1(t) = \sum_{i=1}^{n} \frac{\mid \psi(z,t) - \psi_{ref}(z,t) \mid}{n}$$

and a second order norm error:

$$\varepsilon 2(t) = \sum_{i=1}^{n} \frac{[\psi(z,t) - \psi_{ref}(z,t)]^2}{n}$$

where ψ and ψ_{ref} are the simulated and reference pressure head at time t respectively, for the $i = 1..n$ soil profile points at level z_i. The test problem consists in an infiltration and redistribution simulation into a soil column initially at hydrostatic equilibrium. The boundary condition at the surface is

Table 1
Parameters of the one-dimensional benchmark

θ_r	θ_s	ψ_s, m	n	S_s, m^{-1}	ψ_0, m	K_s, mh^{-1}	Sbc	L, m	Δz, m	Δt, s
0.08	0.54	-3.0	3.0	0.02347	-0.95	5.0	$q = \frac{t}{64}$	10.0	0.1	0.1

a time-varying specified Darcy flux q which increases linearly with time, while the boundary at the base is maintained at a fixed pressure head value of ψ=0, allowing drainage of moisture through the water table. The space grid and time step have been chosen to guarantee the convergence of the system.

Figure 6 shows the comparison between CA simulations and reference numerical solutions: the differences are very small for all the times analyzed, as indicated by the first and second order norm error values shown in table 2.

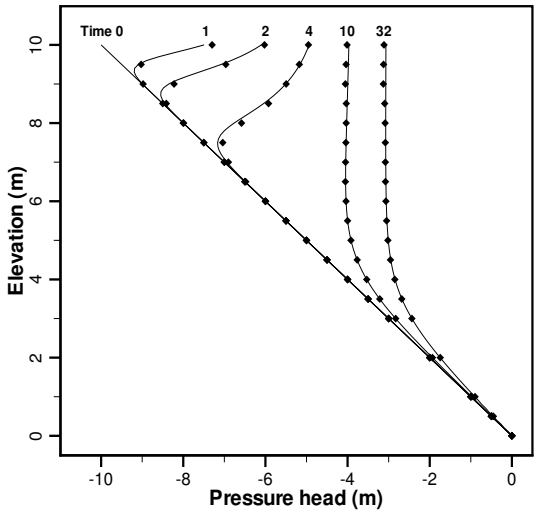

Fig. 6. Comparison between CA simulations (solid lines) and reference numerical solutions (points).

Table 2
First and second order norm error values for the benchmark

Time, h	1	2	4	10	32
$\varepsilon 1$, m	0.0144	0.0189	0.0270	0.0122	0.0155
$\varepsilon 2$, m^2	0.2240	0.1934	0.3361	0.0498	0.0561

The two-dimensional benchmark was based on the physical setting and soil hydraulic properties of the Jornada Test Site near Las Cruces, New Mexico, and was taken from [15]. The test involves transient, two-dimensional infiltration of water into an extremely dry heterogeneous soil, composed by four soil types with different hydrologic properties (table 3). A uniform infiltration rate was used for an area extending 2.25 m from the left boundary of the domain. All boundaries, except for the infiltration zone at the top of the model domain,

were treated as no-flow. The characteristic equations considered are based on the van Genuchten formulation [19] of Mualem [13] model. The relative soil saturation for the entire cross-section obtained at 30 days after the start of moisture infiltration was compared with the results given by the finite difference computer code PORFLOW [9], considered as the reference numerical solutions. Figure 7 shows that the two sets of data compare very well with each other both qualitatively and quantitatively.

Table 3
Parameters of the two-dimensional benchmark

Zone	θ_s	θ_r	α, m^{-1}	n	K_s, md^{-1}
1	0.368	0.1020	3.34	1.982	7.909
2	0.351	0.0985	3.63	1.632	4.699
3	0.325	0.0859	3.45	1.573	4.150
4	0.325	0.0859	3.45	1.573	41.50

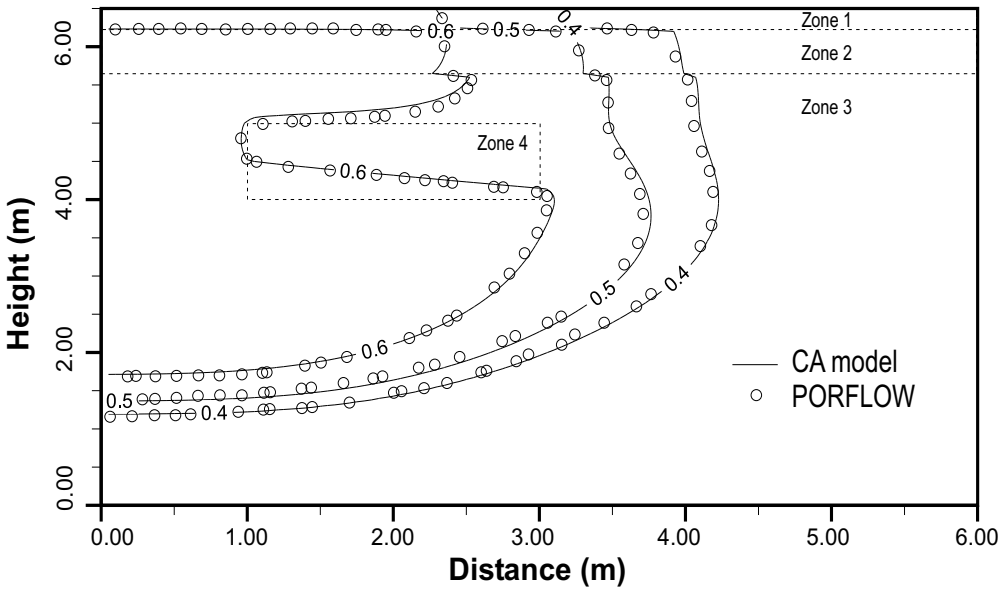

Fig. 7. Comparison between CA simulations (solid lines) and reference numerical solutions (points).

5.2 Validation of the performance model

To validate the use of our scalability prediction technique (subsection 4.1) and to assess its accuracy, some experimental evaluations are carried out using a three-dimensional benchmark, obtained by considering the two-dimensional benchmark as a section of a three-dimensional plot, with a variable depth of z m. Initial and boundary conditions are not modified, but the infiltration area, still extending 2.25 m from the left boundary of the domain, is hypothesized

having a depth of 0.50 m ahead and 0.50 m behind the central value along the third dimension.

For the problem described we considered three different dimensions of the automata: $64 \times 160 \times 150$, $128 \times 160 \times 150$ and $512 \times 160 \times 150$. Icarus, a parallel cluster with 16 nodes, each having two-processors Pentium 1.133 Ghz, 2 Gigabytes of memory Ram and Linux Operating System has been used in the experiments.

Table 4 shows the estimated values of the parameters necessary to evaluate the parallel execution time T_p for different number of processors using equation 9. The $t_f^{''}$ value was estimated by measuring its computational cost for different problem sizes and then using the Matlab toolkit to automatically calculate the least-squares fit of the equation that defines $t_f^{''}$ with the experimental data. t_{ap} was estimated by measuring its computational cost for different number of processors and then calculating the least-squares fit of the equation that defines t_{ap} with the experimental data. The t_b values for the Icarus machine was estimated using a ping-pong test.

Table 4
Estimated problem parameters

Parameter	**Value(μsec)**
$t_f^{''}$	1.544
t_{ap}	715.62
dt_b	1.252

Tables 5, 6 and 7 show the measured and predicted execution times, and the relative error associated with each prediction for different dimensions of the automata.

Table 5
Execution time predictions for 100 iterations ($64 \times 160 \times 150$)

Num. procs	**Num. cells per node**	**Measured (sec)**	**Predicted (sec)**	**Relative Error (percent)**
1	1536000	-	-	-
2	768000	129.58	130.67	0.85%
4	384000	70.22	71.39	1.66%
8	192000	42.65	41.74	-2.14%
16	96000	27.97	26.92	-3.76%
32	48000	20.49	19.51	-4.80%

The results show a good agreement between the model and the experiments. In fact, the measured times were, in the worst case, about 2% for the largest

Table 6
Execution time predictions for 100 iterations ($128 \times 160 \times 150$)

Num. procs	**Num. cells per node**	**Measured (sec)**	**Predicted (sec)**	**Relative Error (percent)**
1	3072000	-	-	-
2	1536000	248.99	249.25	0.10%
4	768000	132.66	130.67	-1.50%
8	384000	73.35	71.39	-2.68%
16	192000	43.62	41.74	-4.31%
32	96000	29.00	26.92	-7.18%

Table 7
Execution time predictions for 100 iterations ($512 \times 160 \times 150$)

Num. procs	**Num. cells per node**	**Measured (sec)**	**Predicted (sec)**	**Relative Error (percent)**
1	12288000	-	-	-
2	6144000	964.16	960.70	-0.36%
4	3072000	489.91	486.40	-0.72%
8	1536000	251.77	249.30	-0.98%
16	768000	132.02	130.70	-1.00%
32	384000	72.77	71.40	-1.88%

dimension and about 5% and 7% for the other two dimensions. Since it is the most accurately estimated model term, the prediction becomes increasingly accurate with larger problems. A more accurate prediction model is obtainable using a refined model of communication cost [4].

From equation 15, we calculate that the value of the speedup is bound to 20 for the first case and to 39 and 157 respectively for the second and third case. The lower value of speedup for the first case is due to the much larger communication/computation ratio. We can obtain a better value of speedup increasing the granularity, that is, allocating a larger number of cells for node. We can use formula 14 to calculate the exact size of A to obtain a desired efficiency. For example, from this formula, maintaining B=160 and C=150 we obtain a size of A equal to 592 to have an efficiency of 85% on 32 processors. The model can be helpful to calculate the correct size of automata in order to obtain a given efficiency for a specific architecture. Furthermore, we can determinate, for a specific dimension, the optimal number of processors that permits to reach a specific efficiency.

With the aim of testing the scalability of the proposed CA model, we simulated by CAMELot, on the Icarus parallel machine, three different models characterized by size equal to 64 × 160 × 150, 128 × 160 × 150, 512 × 160 × 150 respectively, each running with a different number of processors (1, 2, 3, 4, 8, 16, 32). The simulations are summarized in figure 8 where speedup values achieved for each automata dimension, changing the number of utilized processors, are shown. The same figure shows that the larger dimension of automata considered for the simulations is sufficient to reach a scalability very close to the optimal with the maximum number of processors available. The same analysis of the computation-process scalability shows that with the lowest automata resolution (1,536,000 cells) an acceptable efficiency (see table 8) can only be reached using up to 8 processors (about 70%), and therefore a higher number appears unsuitable. This is due to computing times necessary to run the same automata with 16 or 32 processors, which are not capable of balancing the communication times of the parallel system. Instead, for higher automata resolutions, suitable efficiency values are observed increasing the number of processors to 32 (about 82%) which justify their use.

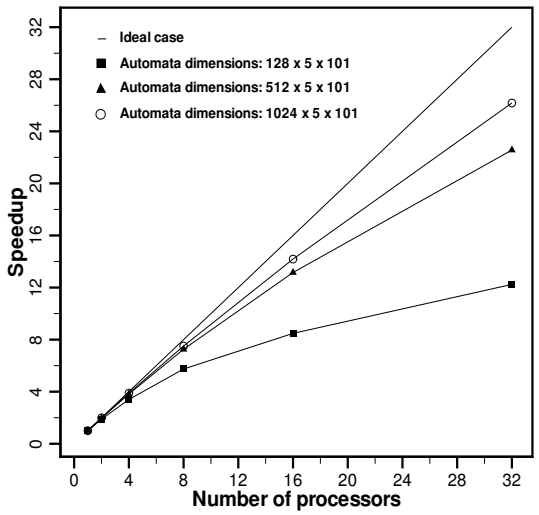

Fig. 8. Comparison between CAMELOT speedup values obtained through the CA model for different automata sizes and the ideal speedup.

6 Quantized model

From equation 10 and 15 it can be noticed that the speedup, and consequently the efficiency, is limited from the time necessary for exchanging mes-

Table 8
Efficiency values for different automata sizes

Number of processors	Automata dimensions $64 \times 160 \times 150$	Automata dimensions $128 \times 160 \times 150$	Automata dimensions $512 \times 160 \times 150$
1	100.00%	100.00%	100.00%
2	92.39%	96.00%	98.98%
4	85.25%	90.09%	97.40%
8	70.17%	81.47%	94.77%
16	53.50%	68.50%	90.36%
32	36.52%	51.52%	81.97%

sages among processors (about $4BCdt_b$). In practice, an entire soil column of size BC exchanges messages with the neighbors. Reducing the number of cells that exchange messages would have a beneficial effect on the speedup.

To this aim, we introduced a quantization strategy, based on the quantization systems theory [22]. Such a process, called "quantization", starting from the discretization of the states of a continuous process, fixes their evolution only through multiples of a specific value, called "quantum size". In a CA system local interactions within the neighborhood, carried out through the transition function, involve each *fa* in temporal changes of the state: if during a time step the application of the transition function does not allow the cell to evolve from state D at least to state $D \pm 1$ (where the difference between two states is given by the quantum size), it maintains its current state, and does not exchange information with the neighborhood. The quantization makes cellular automata asynchronous, because for each iteration each *fa* decides, with respect to the value of its state and those of adjacent cells, to be updated or to remain "frozen" at the previous time step.

For the proposed model the quantization procedure was applied to the main flow parameter represented by the total head gradient. Specifically, if the head difference between two neighbors *fa* is not greater than a given threshold, then the same automata are hypothesized as being at rest. The threshold represents the quantum size and can be static or dynamic: the former is given by a constant value H_q on the whole automata; the latter instead depends on the *fa* state during the simulation. The dynamic threshold has been assumed to be inversely proportional to the *fa* saturation degree, in a form like this:

$$quantum = k\frac{\theta_{sat}}{\theta} \tag{16}$$

The use of dynamic quantization extends the CA model, which becomes both

asynchronous and non-uniform: the transition function changes in each *fa*, because in each automaton the threshold depends on a local factor represented by the degree of saturation. It seems to be that the *fa* decides each time, according to its degree of saturation, what the associated threshold value should be. However, increasing the threshold value produces two opposite effects: on the one hand the number of messages exchanged decreases and on the other the model performs less well.

We have applied the quantization techniques for the solution of the one-dimensional benchmark, with the aim of investigating the effects on the model both in terms of number of messages exchanged and in terms of error of the model. We have measured the error, when the value of static threshold H_q is increased, normalized with regard to a cell column.

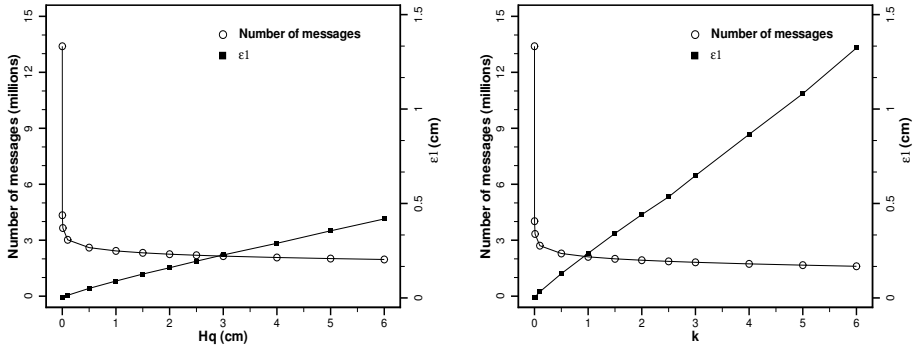

Fig. 9. Reduction in exchanged messages and variation of error norm $\varepsilon 1$ values for the one-dimensional benchmark, at a time of 4 hours, varying (a) static threshold H_q values and (b) dynamic threshold parameter k values.

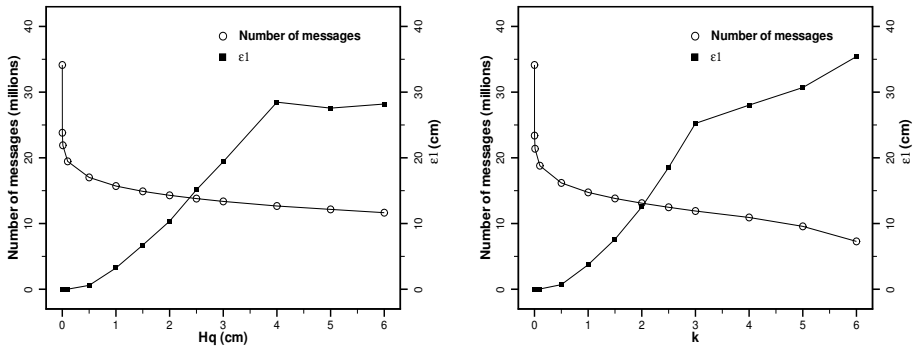

Fig. 10. Reduction in exchanged messages and variation of error norm $\varepsilon 1$ values for the one-dimensional benchmark, at a time of 10 hours, varying (a) static threshold H_q values and (b) dynamic threshold parameter k values.

Figures 9, 10 and 11 (a and b) show the reduction in the number of messages exchanged along a column of cells together with the norm error $\varepsilon 1$ values (in

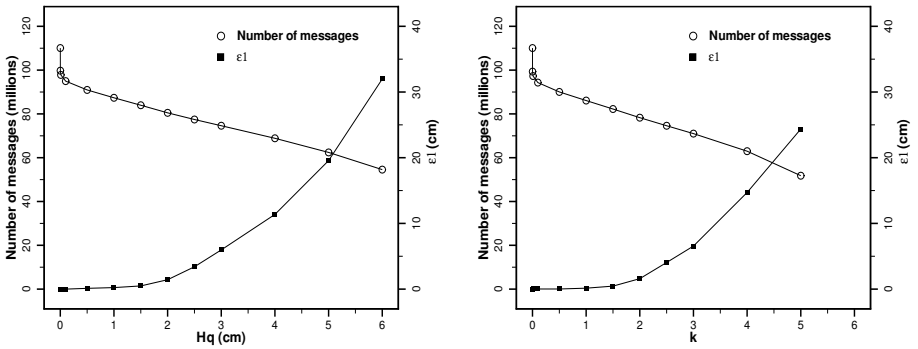

Fig. 11. Reduction in exchanged messages and variation of error norm $\varepsilon 1$ values for the one-dimensional benchmark, at a time of 32 hours, varying (a) static threshold H_q values and (b) dynamic threshold parameter k values.

which the reference values are obtained from the simulation without quantization), when the values of static threshold H_q and dynamic threshold parameter k are changed, respectively at times of 4, 10 and 32 hours. Furthermore, both for static and dynamic thresholds, the percentage reduction in exchanged messages at the same simulation times are shown in table 9. Obviously, messages exchanged for H_q=k=0 are the same.

For all the times analyzed, starting from a very small threshold value (H_q and k equal to 0.01), an appreciable reduction in the number of exchanged messages was obtained. However, this appeared less significant for increasing simulation times. For threshold values greater than 0.001, against a constant increase in the norm error, the further reduction in the messages exchanged is less evident, specially during the first hours of simulation. The reduction in exchanged messages varying the thresholds as well as specific effects produced by static and dynamic approaches have not to be thought of as general results, but are associated to the problem analyzed.

Dynamic threshold has a better behavior (considering the same number of messages, it has a lower error) but this improvement is more evident after many hours of simulation. However, using dynamic threshold, the error is greater; in fact, with k=6 simulation diverges after 12 hours.

7 Conclusions

We have presented a discrete model for unsaturated flow that is particulary suitable to be used with the CA paradigm in a parallel computing environment. Based on this CA model, 3D parallel simulations have been carried out by the CAMELot environment. We have demonstrated the accuracy of the

Table 9
Percentage reduction in exchanged messages at times of 4, 10 and 32 hours, both for static and dynamic threshold

H_q / k	Static threshold	Dynamic threshold	Static threshold	Dynamic threshold	Static threshold	Dynamic threshold
0	-	-	-	-	-	-
0.001	67.61	70.00	30.22	31.48	9.37	9.76
0.01	72.74	75.11	35.81	37.36	11.13	11.61
0.1	77.45	79.79	42.99	44.91	13.72	14.32
0.5	80.57	82.92	50.15	52.59	17.37	18.20
1.0	81.89	84.26	54.00	56.83	20.63	21.77
1.5	82.66	85.05	56.35	59.52	23.72	25.29
2.0	83.20	85.62	58.12	61.63	26.87	28.89
2.5	83.63	86.07	59.59	63.46	29.64	32.24
3.0	83.97	86.45	60.84	65.13	32.22	35.49
4.0	84.52	87.07	62.89	68.01	37.43	42.79
5.0	84.96	87.58	64.41	71.96	43.29	52.96
6.0	85.31	88.04	65.87	78.68	50.43	

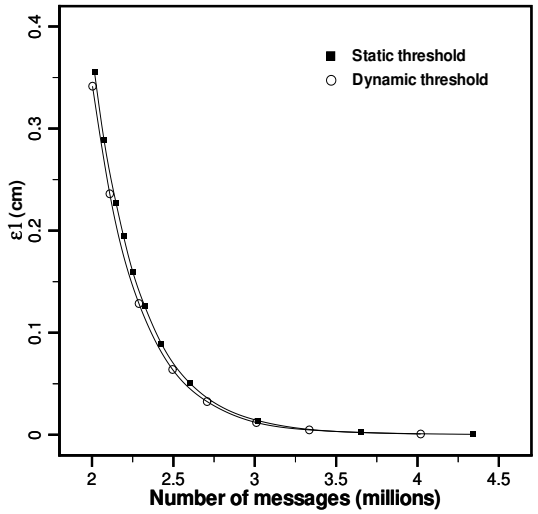

Fig. 12. Comparison between static and dynamic threshold simulations plotting the number of exchanged messages vs. the norm error values at a time of 4 hours.

CA model by two significant benchmarks. Also, we have analyzed the scal-

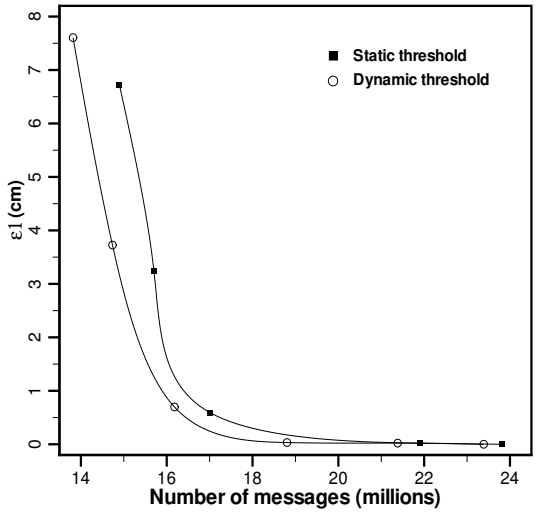

Fig. 13. Comparison between static and dynamic threshold simulations plotting the number of exchanged messages vs. the norm error values at a time of 10 hours.

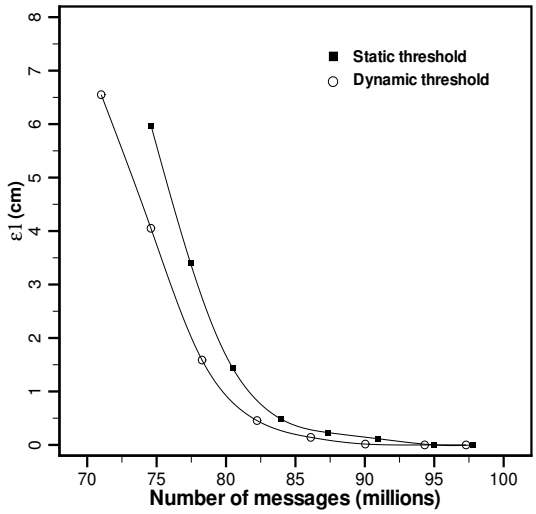

Fig. 14. Comparison between static and dynamic threshold simulations plotting the number of exchanged messages vs. the norm error values at a time of 32 hours.

ability of the model using an analytical performance model. The model has been evaluated using a three-dimensional benchmark. Our preliminary results showed a good agreement between the model and the experiments. To further improve the performance on the model a quantized version has been carried out. Results have shown that the quantization techniques increase the performance of the model reducing the execution time. Finally, this study has shown that CA are a useful paradigm for modeling and simulation of unsaturated flow processes and CAMELot is a flexible and a powerful tool for simplifying

the construction of new models and easing their verification.

References

[1] P.M. Adler. *Multiphase flow in porous media*. Kluwer, 1995.

[2] J. Bear. *Dynamic of fluid in porous media*. American Elsevier, New York, 1972.

[3] G. Dattilo and G. Spezzano. Simulation of a cellular landslide model with camelot on high performance computers. *Parallel Computing*, 29(10):1403–1418.

[4] I. Foster. *Designing and Building Parallel Programs*. Addison-Wesley, 1995.

[5] U. Frisch, D. d'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.P. Rivet. A lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1:649–707, 1987.

[6] A. Y. Grama, A. Gupta, and V. Kumar. Isoefficiency: measuring the scalability of parallel algorithms and architectures. *IEEE parallel and distributed technology: systems and applications*, 1(3):12–21, August 1993.

[7] S. Di Gregorio and R. Serra. An empirical method for modelling and simulating some complex macroscopic phenomena by cellular automata. *Future Generation Computer Systems*, 16(2-3):259–271, 1999.

[8] R. W. Hockney. The communication challenge for mpp: Intel paragon and meiko cs-2. *Parallel Computing*, 20(3):389–398, 1994.

[9] S.O. Magnuson, R.G. Baca, and A.J. Sondrup. Independent verification and benchmark testing of the porflo-3 computer code, version 1.0. Technical Report EGG-BG-9175, Idaho National Engineering Laboratory, Idaho Falls, ID, 1990.

[10] G. Manzini and E. Ferraris. Mass-conservative finite volume methods on 2-d unstructured grids for the richards equation. *Advances in Water Resources*, 27:1199–1215, 2004.

[11] C. Mattiussi. An analysis of finite volume, finite element, and finite difference methods using some concepts for algebraic topology. *Journal of Computational Physics*, 133:289–309, 1997.

[12] G. Mendicino, A. Senatore, G. Spezzano, and S. Straface. Three-dimensional unsaturated flow modelling using cellular automata. *submitted to Water Resoucers Research*, 2005.

[13] Y. Mualem. A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resour. Res.*, 12:513–522, 1976.

[14] C. Paniconi, A.A. Aldama, and E.F. Wood. Numerical evaluation of iterative and noniterative methods for the solution of the nonlinear richards equation. *Water Resour. Res.*, 27(6):1147–1163, 1991.

[15] J.D. Smyth, S.B. Yabusaki, and G.W. Gee. Infiltration evaluation methodology - letter report 3: Selected tests of infiltration using two-dimensional numerical models. Technical report, Pacific Northwest Laboratory, Richland, WA, 1989.

[16] G. Spezzano and D. Talia. The carpet programming environment for solving scientific problems on parallel computers. *Parallel and Distributed Computing Practices*, 1(3):49–61, 1998.

[17] S. Succi, R. Benzi, and F. Higuera. The lattice boltzmann equation: a new tool for computational fluid dynamics. *Physica*, 47(D):219–230, 1991.

[18] E. Tonti. A discrete formulation of field laws: the cell method. *Computer Modeling in Engineering and Science*, 2(2):237–258, 2001.

[19] M.Th. van Genuchten. Calculating the unsaturated hydraulic conductivity with a new closed-form analytic model. Water Resources Program Report 78-WR-08, Department of Civil Engineering, Princeton University, Princeton, NJ, 1978.

[20] M.Th. van Genuchten and D.R. Nielsen. On describing and predicting the hydraulic properties of unsaturated soils. *Ann. Geophys.*, 3(5):615–628, 1985.

[21] S. Wolfram. *Cellular Automata and Complexity: Collected Papers*. Addison-Wesley, 1994.

[22] B.P. Ziegler. Devs theory of quantized systems. Technical report, Advanced simulation technology thrust DARPA contract, 1998.