Journal of Computational Physics 229 (2010) 6450–6465

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A numerical simulation method for dissolution in porous and fractured media

D. Yu, A.J.C. Ladd *

Department of Chemical Engineering, University of Florida, Gainesville, FL 32611-6005, USA

ARTICLE INFO

Article history:
Received 9 October 2009
Received in revised form 7 May 2010
Accepted 11 May 2010
Available online 19 May 2010

Keywords:
Dissolution
Porous media
Lattice-Boltzmann
Finite-difference

ABSTRACT

We describe an algorithm for simulating reactive flows in porous media, in which the pore space is mapped explicitly. Chemical reactions at the solid–fluid boundaries lead to dissolution (or precipitation), which makes it necessary to track the movement of the solid–fluid interface during the course of the simulation. We have developed a robust algorithm for constructing a piecewise continuous (C_1) surface, which enables a rapid remapping of the surface to the grid lines. The key components of the physics are the Navier–Stokes equations for fluid flow in the pore space, the convection–diffusion equation to describe the transport of chemical species, and rate equations to model the chemical kinetics at the solid surfaces. A lattice-Boltzmann model was used to simulate fluid flow in the pore space, with linear interpolation at the solid boundaries. A finite-difference scheme for the concentration field was developed, taking derivatives along the direction of the local fluid velocity. When the flow is not aligned with the grid this leads to much more accurate convective fluxes and surface concentrations than a standard Cartesian template. A robust algorithm for the surface reaction rates has been implemented, avoiding instabilities when the surface is close to a grid point. We report numerical tests of different aspects of the algorithm and assess the overall convergence of the method.

1. Introduction

The transport of reactants in porous media is a fundamental geochemical problem [1,2], with growing relevance to sequestration of excess carbon dioxide [3,4]. Aqueous CO_2 is a weak acid which gradually erodes alkaline mineral formations, particularly limestone. Fluid flow through the porous matrix introduces carbonic acid (H_2CO_3), which dissolves the surrounding solid. However the increase in porosity is non-uniform and under some conditions the flow gradually becomes focused into a small number of long channels [5]. Flow focusing is a possible mechanism underlying the formation of limestone caverns [6] and is also important in estimating geological confinement times for stored CO_2 [7]. Most numerical work in modeling dissolution of porous media has been at the Darcy scale [8,9], while similar models, based on a two-dimensional Reynolds approximation, have been applied to fracture dissolution [7,10]. However it is now becoming feasible to incorporate detailed microscopic pore structures [11,12], particularly with the growing use of lattice-Boltzmann methods [13–15], which are well suited to complex geometries. In this paper we propose a number of algorithmic improvements, and in particular a more accurate representation of the interface between the solid matrix and the fluid.

The simplest representation of a porous material is to divide the system into cells, with each cell being either solid or fluid [14,15]. However, the accuracy of such a scheme is limited, unless a very large number of grid points are used. Such simulations have so far been limited to two-dimensional systems [16]. We have previously developed a lattice-Boltzmann model

* Corresponding author. Fax: +1 652 392 9513.
E-mail address: ladd@che.ufl.edu (A.J.C. Ladd).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.05.005

that could account for intermediate grid positions [17] and thus allow three-dimensional simulations to be carried out [13]. More recently, we demonstrated that a lattice-Boltzmann flow solver [18] and a random-walk algorithm for the reactant transport [19,20] can simulate erosion on the scale of laboratory experiments [21], using either numerically generated [22,23] or experimentally measured [21] topographies. Nevertheless, the method suffers from two significant limitations.

First, the random-walk algorithm required a particularly efficient method for calculating the erosion fluxes [20], which eliminated the statistical noise from fluctuations in the local concentration at the solid surface [21]. However, this algorithm is limited to linear kinetics, but it is known that the dissolution of limestone by aqueous CO_2 is strongly non-linear, with a substantial reduction in reaction rate near saturation. A finite-difference method has the advantage that there is no statistical error and it can be extended to non-linear kinetics. We use finite-difference, in preference to lattice-Boltzmann [24,25,15,26] and related methods [27,28], in order to utilize upwind differencing at high Péclet numbers. Lattice-Boltzmann methods for multicomponent transport have not yet implemented upwind differencing, and are thus restricted to relatively low Péclet numbers. Recent work [26,29,28] has extended the range of Péclet numbers that can be investigated by lattice-Boltzmann methods, but when concentration gradients are sharp it is difficult to obtain accurate solutions when the grid Péclet number exceeds 30, which is typically the range where a finite-difference scheme will switch to upwind differencing. By contrast, simulations of fracture flows can require a grid Péclet number in excess of 100 in order to achieve a computationally practical number of grid points. The primary concern with a grid-based method is the resolution needed to represent the concentration field in sharply varying geometries. We have therefore investigated the convergence of the transport solver in complex topographies, characteristic of porous media.

The other limitation of our previous work [21,23] is that the surface was represented by two height maps, defining the z position of each surface as a function of the x,y coordinates. This makes the erosion of the surface much easier to calculate, but it cannot accurately account for the dissolution of rough topographies, where there may be substantial variations in height over scales comparable to the pore size. Here we have implemented a full three-dimensional surface representation, which is robust even with a sharply varying topography. The description of the surface and the algorithm for updating it are one of the key innovations of this work. The other innovation is an improved finite-difference solver, which leads to accurate surface concentrations in situations where the standard differencing template does not. The method is fully parallelized and runs efficiently on a small cluster of up to 100 processors, connected by gigabit Ethernet. The algorithm is described in the next section, Section 2.

In Section 3 we describe a number of numerical tests of the algorithm. We first present results for the permeability of a dense random array of spheres, which is used to test the accuracy of the flow solver. Next we calculate the concentration field in a narrow channel, both aligned with the grid and at an angle to the grid. We compare results for different implementations of the finite-difference solver with a reference solution obtained with programs from the NAG library [30]. The results demonstrate the improved accuracy of our finite-difference scheme, which is related to the method of characteristics. We then illustrate the precision of the surface-update algorithm, using a uniformly expanding and contracting sphere. Finally, we examine erosion patterns in an artificial fracture, in order to demonstrate the convergence of the method with increasing resolution of the flow and concentration fields.

2. A computational algorithm for dissolution

A simulation of dissolution in a porous matrix requires three main computational components: fluid flow, reactant transport, and surface erosion. We solve the Navier–Stokes equations in the weak compressible form using a lattice-Boltzmann model (Section 2.4):

$$\partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{1}$$

$$\rho \partial_t \boldsymbol{u} + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} + c_s^2 \nabla \rho = \eta \nabla^2 \boldsymbol{u} + \left(\zeta + \frac{\eta}{3}\right) \nabla \nabla \cdot \boldsymbol{u}. \tag{2}$$

Here ρ and $\boldsymbol{u}$ are the fluid mass density and velocity, c_s is the speed of sound, and η and ζ are the shear and bulk viscosities. At sufficiently low Mach numbers, $M = u/c_s \sim 0.1$, these equations are a good approximation to the incompressible Navier–Stokes equations [31].

Next, the distribution of reactants (and possibly products) must be determined from a solution of the convection–diffusion equation:

$$\partial_t c + \boldsymbol{u} \cdot \nabla c = D \nabla^2 c, \tag{3}$$

using the flow field determined by the lattice-Boltzmann simulation. A similar equation can be solved for every species, but here we consider only a single component, with diffusion constant D. A new approach to discretizing the convection–diffusion equation is given in Section 2.5.

There are separate grids for the fluid solver and concentration solver, since the resolutions required for each may be different. Typically the Péclet number $Pe = Q/AD$ is larger than the Reynolds number, $Re = Q/Av$; here Q is the volumetric flow rate, A is the cross-sectional area of the porous matrix, and $v = \eta/\rho$ is the kinematic viscosity of the fluid. Thus although there is frequently a thin concentration boundary layer, the fluid velocity varies slowly across the pore space.

The chemical kinetics at the solid surface serve as a boundary condition on Eq. (3); for a single species:

$$D\boldsymbol{n}\cdot(\nabla c)_0 = R(c_0), \tag{4}$$

where the surface normal, $\boldsymbol{n}$, points into the fluid and the subscript 0 indicates a position on the surface. The reaction rate, $R(c_0)$, is negative for dissolution, and for linear kinetics is proportional to the undersaturation:

$$R(c) = k(c_0 - c_{\mathrm{sat}}), \tag{5}$$

where k is the rate constant and c_{sat} is the saturation concentration.

Having calculated the fluxes at the solid–fluid surface, the topography of the solid matrix is updated (Section 2.3). In the applications of interest to us [22,23], the rate of dissolution of the solid is very slow and both the fluid and transport equations can be assumed to be at steady state for each specific topography. Here we describe the algorithm in these terms, separately cycling through the flow solver until the velocity field is stationary in time, then iterating the convection–diffusion equation to steady state, and finally calculating the erosion velocity and updating the surface. However the code can also be run in fully coupled mode if necessary, for example to simulate fast dissolution by strong acids.

2.1. Representing the solid–fluid interface

The boundary between the solid and fluid phases is described by a set of marker points, which denote all the positions where grid lines intersect the solid–fluid interface. The initial positions of the marker points are determined by interpolating a piecewise continuous (C^0) representation of the surface. An example of the labeling of the nodes is shown in Fig. 1. Based on the initial position of the surface, nodes are labeled as solid (open squares) or fluid (open circles) depending on which side of the interface they are on: Fig. 1 indicates a convex solid surface. Once the solid and fluid nodes are assigned, the neighborhood of each solid node is checked along the Cartesian directions. If fluid nodes are found then the node in question is labeled as a solid–fluid boundary node (solid squares). The distances to the surrounding marker points (solid diamonds) are recorded for each boundary node; there is at least such one distance for each boundary node and no more than six. Note that the marker points are not explicitly defined in the code; the information about their location is stored in the solid boundary node data structure.

The code allows for different grid resolutions in the fluid solver (coarse grid) and concentration solver (fine grid). This is useful because there is frequently a concentration boundary layer near the solid surface, while at low Reynolds number the fluid velocity varies more smoothly. The geometry of the solid–fluid interface is defined on the fine grid but the solid boundary nodes on the coarse grid must be determined separately, as illustrated in Fig. 2. The flow solver uses the D3Q19 lattice-Boltzmann model, in which additional distances along diagonal directions between each pair of coordinate axes are needed (Fig. 2). The marker points for the diagonal directions are placed by linear interpolation between cardinal points (see Fig. 2b).

The node map and distances to the marker points are the only geometric data needed by the fluid and concentration solvers. The connections between markers are not needed as long as the surface remains stationary. However the markers move as a result of erosion and connectivity information is needed to calculate the new marker positions. We use a local representation of the surface in terms of triangular Bezier functions, as described in Section 2.3. We have also implemented a marching cube algorithm [32] to calculate the fluid fraction surrounding each grid point, which is used to determine the local fluid velocity (Section 3.1).

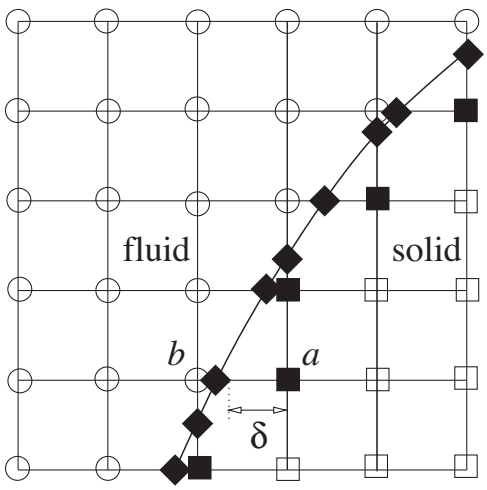

Fig. 1. Solid–fluid surface configuration. The various node types are indicated by different symbols; fluid nodes (open circles), solid nodes (open squares), solid boundary nodes (filled squares) and surface markers (filled diamonds).

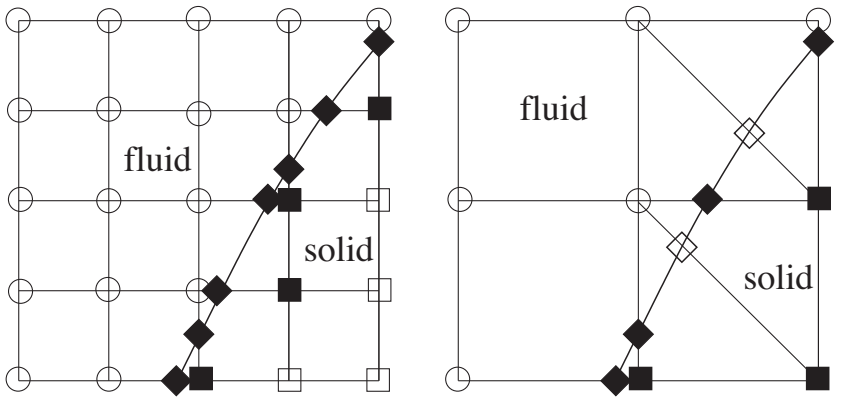

Fig. 2. Mapping the geometry from the fine grid to the coarse grid. Note that some solid nodes on the fine grid can turn to solid boundary nodes on the coarse grid. The additional marker points for the diagonal intersections (open diamonds) are found by interpolating between the cardinal points.

2.2. Calculating the surface normals

Surface normals are needed at the marker points, both to determine the direction of the erosion flux, Eq. (4), and to construct third-order Bézier surfaces. A planar triangulation of the surface requires only the coordinates, but a C^1 surface needs the normals as well; they can be calculated from the vector product of tangents to the surface at each marker point. The tangents do not need to be perpendicular to each other, but the accuracy with which the normal can be calculated is degraded if they are nearly parallel.

A suitable pair of tangents can be determined from four neighboring points around the marker, taken along directions normal to the link connecting the solid boundary node and the surface marker point, as illustrated in Fig. 3. In this case, the marker point lies along the x axis and suitable neighboring points can be found along the y and z directions. Fig. 3 illustrates the search for a single neighbor in the positive y direction, checking the state of grid points around the marker. For example, if the nodes directly above it (nodes c and d) are solid, then there is a surface marker (1) somewhere along the link $b - d$, which is taken as the neighbor. On the other hand, if c is solid and d is fluid the surface intersects with $c - d$ and point 2 is taken. A different algorithm is used for $\delta < 0.5\Delta x$ and $\delta > 0.5\Delta x$, which avoids using two nearby points to compute a tangent. The complete algorithm is described in the flow chart shown in Fig. 4.

The neighboring point in the negative y direction is found in a similar fashion. Three markers, $d - a - b$ in Fig. 5, are used to determine a tangent at the marker point a, by fitting a circle to their positions. The process is repeated for neighbors in the remaining (z) direction, which in Fig. 5 is the point c and an unseen point in front. Finally the vector product of the tangents is taken to obtain the normal at a. The method is robust; it always finds neighboring points, regardless of the configuration of solid and fluid nodes. This is advantageous when the geometry is highly irregular.

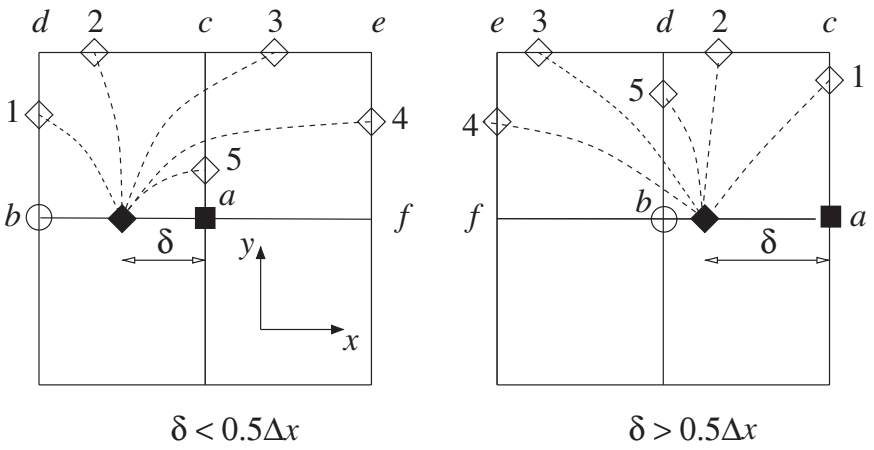

Fig. 3. Construction of a surface tangent at the marker point (solid diamond) defining the intersection of the surface with the grid line (x direction) connecting the fluid point b (open circle) with the solid boundary point a (filled square). The figure illustrates the method for finding a single neighbor in the positive y direction from the five candidate locations (open diamonds), depending on the state of the nodes $c - f$ (fluid or solid). Cases where $\delta < 0.5$ and $\delta > 0.5$ are treated separately, to avoid using nearby points.

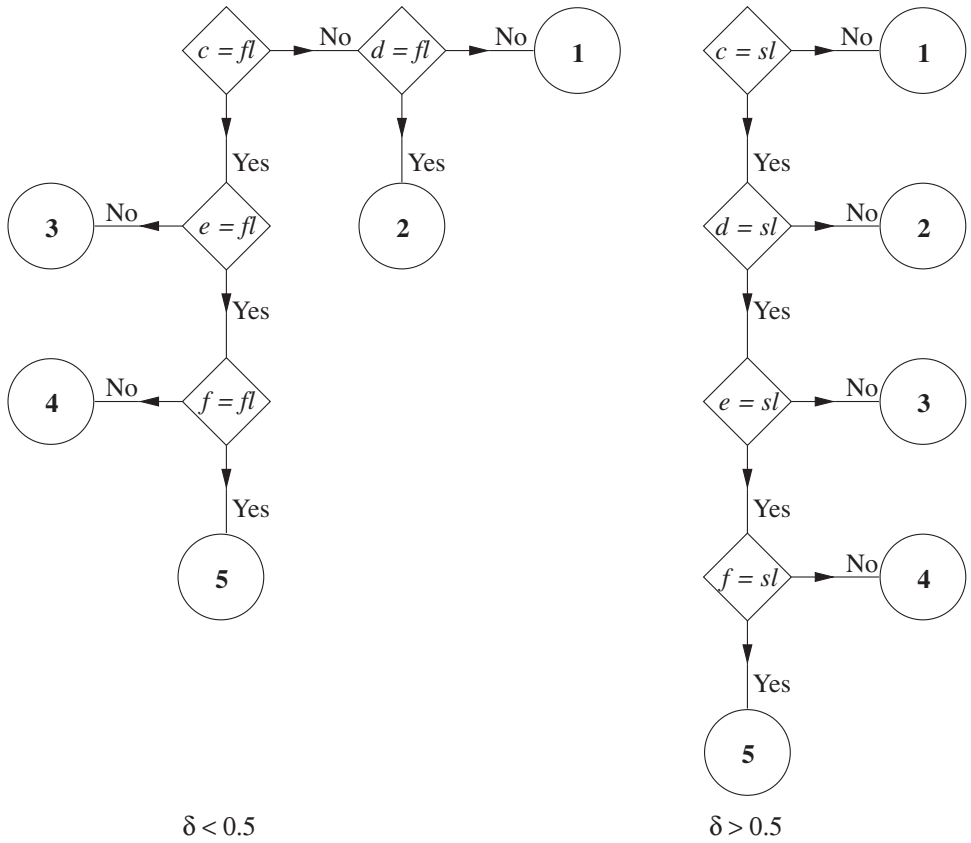

Fig. 4. Neighbor-finding algorithm illustrated in Fig. 3. The flowchart describes the algorithm for finding a single neighbor in the positive y direction from the five candidate locations, depending on the state of the nodes $c - f$; fluid (*fl*) or solid (*sl*). Cases where $\delta < 0.5$ and $\delta > 0.5$ are treated separately.

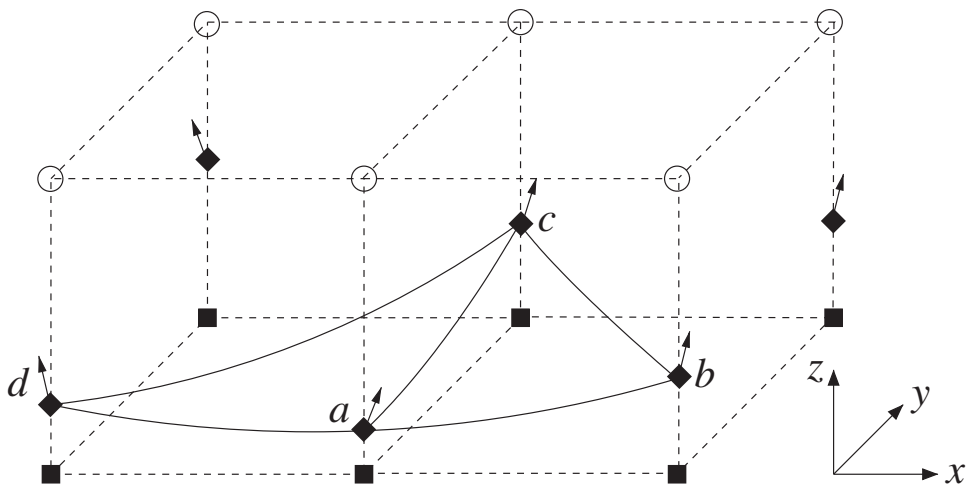

Fig. 5. Construction of local tangents. The figure shows two grid cells containing solid boundary nodes (solid squares) and surface markers (solid diamonds). The surface normals at the marker points, pointing into the fluid, are indicated by arrows.

2.3. Moving the marker points

Erosion of the surface is described by motion of the marker points. In a small time increment, Δt, a marker moves a distance $v\Delta t$ in the direction of the surface normal:

$$\boldsymbol{v} = \frac{\nu^{aq} R(c_0)}{\nu^{sl} c^{sl}} \boldsymbol{n}, \tag{6}$$

here ν^{aq} and ν^{sl} are the stoichiometric numbers in the aqueous and solid phases, and c^{sl} is the concentration in the solid phase [23]. The concentration at the marker point is determined from the boundary conditions on the convection–diffusion equation, (4); details of the implementation of the boundary condition are given in Section 2.6. Since there is a wide variation in erosion rates, we have found it more efficient to set a maximum erosion depth for each cycle and then determine the corresponding Δt.

After the markers have moved, a representation of the surface around each marker point is needed to calculate the intersections of this new surface with the grid lines (Fig. 6). A C^1 continuous Bézier surface can be constructed around a central point from four neighboring markers and normals. The neighboring markers found during the construction of the surface normal (Fig. 5) are reused, but in their new positions; three of the four neighboring points of marker a are shown in Fig. 6 (b, c, d) in new locations. Since the markers move along the direction of the surface normals, we keep the previously computed normals.

Fig. 6 illustrates the calculation of a new intersection point (open diamond) of an existing marker, a, which lies along a grid line parallel to the z axis. The marker point and two of its neighbors (for example abc or acd) are projected onto the xy plane (normal to the grid line) and their locations in the plane are used to calculate the barycentric coordinates u, v, and w for that triangle [33]. Then it is straightforward to tell if the grid line intersects the triangle; in Fig. 6 it intersects with abc. The new position of the marker is then calculated from the Bézier function of that triangle [33]:

$$\begin{aligned} P(u,v,w) = {} & b_{3,0,0}u^3 + b_{0,3,0}v^3 + b_{0,0,3}w^3 + 3\left(b_{2,1,0}u^2 v + b_{2,0,1}u^2 w + b_{0,2,1}v^2 w + b_{1,2,0}v^2 u + b_{1,0,2}w^2 u + b_{0,1,2}w^2 v\right) \\ & + 6uvw\left(\frac{v^2w^2 b^1_{1,1,1} + w^2u^2 b^2_{1,1,1} + u^2v^2 b^3_{1,1,1}}{v^2w^2 + w^2u^2 + u^2v^2}\right). \end{aligned} \tag{7}$$

The parameters $b_{3,0,0}$, $b_{0,3,0}$, and $b_{0,0,3}$ are the heights of the three vertexes a,b, and c above the plane on which the Bézier triangle is constructed. The parameters $b_{2,1,0}$, $b_{2,0,1}$, $b_{1,2,0}$, $b_{1,0,2}$, $b_{0,2,1}$, and $b_{0,1,2}$ are derived from the slope of the tangent plane at each vertex [33]. The remaining coefficients, $b^1_{1,1,1}$, $b^2_{1,1,1}$, and $b^3_{1,1,1}$ are chosen to satisfy C^1 continuity along the edges of the triangle [33].

The new marker positions may indicate a change in the state of a grid node. For example, the fluid node p in Fig. 6 is converted from a solid boundary node (Fig. 5) by the motion of the marker (a). The distance of each marker from its solid boundary node must therefore be checked after it has moved. A negative distance indicate that the solid boundary node has been converted to a fluid node by the erosion process. Precipitation can cause the reverse transition, a fluid node to a solid boundary node, which is indicated by a distance larger than Δx. When there is a change of state, as in Fig. 6, some markers may be deleted and others added. For example, if the node q remains solid then a new surface marker will be needed along the line

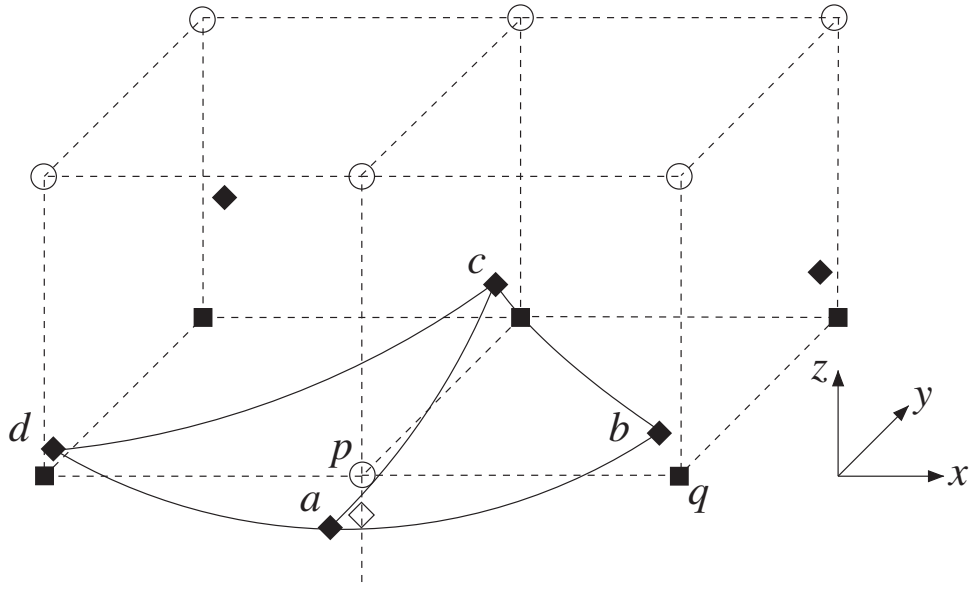

Fig. 6. Reconstruction of surface markers. After erosion, the surface markers (solid diamonds) move to new positions. Two of the four triangles (*abc* and *acd*) comprising the local Bezier surface around marker point *a* are shown; the other two are in front. Intersections between the Bezier surfaces and the grid lines give the new positions of the boundary node markers. The motion of marker *a* has created a new fluid node, *p*, and several new surface points (not shown). However, only the new location (open diamond) of an existing marker is calculated from the Bézier surface.

$p - q$. After reassigning the state of all the affected grid nodes, excess markers are removed and new markers are added. The position of the new marker along the grid line can usually be found by interpolation from neighboring points, for example a and b in the case of the marker along $p - q$. We use cubic interpolation, in this case along a curve in the xz plane, including information from the surface normals at the neighboring markers. If the local curvature is large, it is possible that there is no pair of coplanar neighbors; in this case the new marker is placed at the half-way position.

The algorithm to move the markers requires only four neighboring points to determine the local Bézier surface. Since the motion is along the direction of the surface normal, it is rare that the markers move outside the Bézier surface, but in regions of high curvature this does occasionally happen. The algorithm is made more robust by constraining the maximum distance that any marker can move, typically to less than $0.1\Delta x$. The computational efficiency can be improved by making several updates of the boundary markers without recomputing the flow and concentration fields, but keeping the erosion rate at each marker point fixed. In cases where new markers are introduced the erosion rate is calculated by interpolation. The distance the surface can move between cycles of flow and transport is then limited only by the rate at which the surface concentrations are changing and not by the limitations of the surface-update algorithm.

Alternatively, marching cubes [34,32] could be used to construct a globally self-consistent, triangulated surface at every iteration [35]. Tangents, normals and Bézier surfaces could be constructed as described above, but ensuring that all intersections are found would require a more extensive neighbor search, as well as the overhead from the marching cube algorithm. We have implemented the marching cube algorithm, although at present it is used only to calculate local fluid fractions, as described in Section 3.1.

2.4. Lattice-Boltzmann method

In the lattice-Boltzmann (LB) model, the fluid degrees of freedom are represented by a discretized one-particle velocity distribution function $n_i(\boldsymbol{r},t)$, which describes the mass density of particles with velocity $\boldsymbol{c}_i$ at the lattice position $\boldsymbol{r}$ and time t. The hydrodynamic fields, mass density ρ, momentum density $\boldsymbol{j} = \rho\boldsymbol{u}$, and momentum flux $\boldsymbol{\Pi}$ are moments of this velocity distribution:

$$\rho = \sum_i n_i, \quad \boldsymbol{j} = \sum_i n_i \boldsymbol{c}_i, \quad \boldsymbol{\Pi} = \sum_i n_i \boldsymbol{c}_i \boldsymbol{c}_i. \tag{8}$$

The time evolution of $n_i(\boldsymbol{r},t)$ is described by a discrete analogue of the Boltzmann equation [36]:

$$n_i(\boldsymbol{r} + \boldsymbol{c}_i\Delta t, t + \Delta t) = n_i(\boldsymbol{r}, t) + \Delta_i[\boldsymbol{n}(\boldsymbol{r}, t)], \tag{9}$$

where Δ_i is the change in n_i due to instantaneous collisions at the lattice nodes and Δt is the time step. We used the D3Q19 model [37], with the usual equilibrium distribution:

$$n_i^{\mathrm{eq}}(\rho, \boldsymbol{u}) = a^{c_i}\rho\left(1 + \frac{\boldsymbol{u}\cdot\boldsymbol{c}_i}{c_s^2} + \frac{\boldsymbol{u}\boldsymbol{u} : (\boldsymbol{c}_i\boldsymbol{c}_i - c_s^2\boldsymbol{1})}{2c_s^4}\right), \tag{10}$$

and weights:

$$a^0 = \frac{1}{3}, \quad a^1 = \frac{1}{18}, \quad a^{\sqrt{2}} = \frac{1}{36}. \tag{11}$$

The speed of sound $c_s = 3^{-1/2}\Delta x/\Delta t$, where Δx is the lattice spacing.

A multi-relaxation-time (MRT) collision operator [18,38] was implemented by expanding the non-equilibrium distribution, $n_i^{\mathrm{neq}} = n_i - n_i^{\mathrm{eq}}$, in tensorial polynomials of $\boldsymbol{c}_i$:

$$m_k = \sum_i n_i^{\mathrm{neq}} e_{ki}. \tag{12}$$

We use a different basis ($\boldsymbol{e}_k$) from Refs. [18,38], such that the back transformation includes the weights a^{c_i} [39]:

$$n_i = a^{c_i}\sum_k w_k^{-1} e_{ki} m_k, \tag{13}$$

where $w_k = \sum_i a^{c_i} e_{ki}^2$ is the normalizing factor for $\boldsymbol{e}_k$. The non-equilibrium moments m_k relax towards equilibrium (zero), with a post-collision value:

$$m_k^\star = \gamma_k m_k. \tag{14}$$

A two-parameter collision operator was used, with different eigenvalues for the modes with odd and even powers of $\boldsymbol{c}_i$ [40]. The shear viscosity is related to the eigenvalue (γ) of the even modes:

$$\eta = \frac{\rho c_s^2 h}{2}\left(\frac{1+\gamma}{1-\gamma}\right), \tag{15}$$

while the odd (γ') eigenvalue is chosen so as to make the location of a planar solid boundary independent of γ [39,40]:

$$\gamma' = -\frac{7\gamma + 1}{\gamma + 7}. \tag{16}$$

Linear interpolation [41,42] was used to improve the accuracy of the solid–fluid boundary condition, which requires the distance from the solid node to the surface marker, shown as δ in Fig. 1. The set of distances δ uniquely define the solid surface, without requiring information about the connectivity or shape of the surface. The boundary conditions for each population density, n_i, can then be enforced based solely on the value of δ, with no other information required. This is an important advantage of LB methods over conventional CFD for flow in porous media and one of the reasons for its growing popularity.

2.5. Finite-difference method for scalar transport

Numerical methods for solving the scalar transport equation, Eq. (3), have been extensively studied in the literature, including finite volume, finite-difference, and finite elements [43]. The focus of those methods has been on deriving high-order schemes and avoiding oscillations near boundaries. However, in simulations of porous media, the wide range of length scales implies that there are only a few grid points between opposing surfaces; moreover the irregular geometry means that the fluid flow will not be aligned with the grid. In this work we have adopted a low-order finite-difference scheme, with the aim of moderate fidelity, robustness, and simplicity.

In a discretized Cartesian grid, the convection term is usually calculated in each coordinate direction separately, and the individual fluxes are then added together. For example, in two dimensions, the partial differential equation:

$$\partial_t c + u_x \partial_x c + u_y \partial_y c = D\left(\partial_x^2 c + \partial_y^2 c\right), \tag{17}$$

is solved on a 5-point template using either upwind or centered differencing for the convective flux (depending on the local grid Péclet number) and centered differences for the diffusive flux. However, numerical results (see Section 3.2) show that when the flow is not aligned with the grid there are large errors in the concentration field, particularly if there are source terms in the computational domain. Here, we present a simple algorithm for the convective flux, which leads to much better accuracy than the standard template when the grid is coarse.

The key idea is to calculate the derivative of the concentration along the direction of the local velocity (or characteristic), as illustrated in Fig. 7. The concentrations are found by interpolation to locations c_+ and c_- that lie along the direction of $\boldsymbol{u}$. Then the convective flux is obtained from a single one-dimensional finite-difference. For a centered difference:

$$\boldsymbol{u} \cdot \nabla c = |u| \frac{c_+ - c_-}{d}, \tag{18}$$

while for an upwind scheme:

$$\boldsymbol{u} \cdot \nabla c = 2|u| \frac{c_0 - c_-}{d}. \tag{19}$$

The code switches from a centered difference to an upwind difference at a grid Péclet number, $Pe_c = |u|h/D$, of approximately 30. We have implemented both local and global criteria for upwind differencing. In general we found that a global criterion leads to faster convergence of the iterative (successive over-relaxation) concentration solver, and was used in preference to the local one.

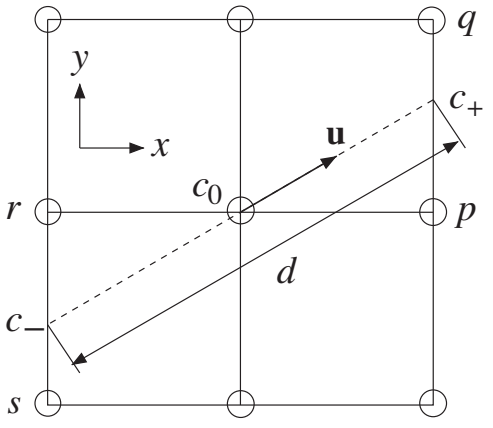

Fig. 7. Calculation of the convective flux by differencing along the flow direction. In a two-dimensional geometry, the concentrations at the grid points labeled p, q, r, and s are linearly interpolated to obtain the concentrations c_+ and c_-.

In the two-dimensional example in Fig. 7, the interpolated concentration c_- can be found from the concentrations at r and s. In three dimensions, the first intersection of the velocity vector with the planes containing the grid lines is found. Then, the concentration at the intersection point is determined by two-dimensional interpolation, using the triangle of grid points surrounding the intersection. Near the solid surface one or more of these points may be missing, and in such cases we take the concentration from the nearest grid point. Since the fluid velocity is largely parallel to the solid surface, such intersections are rare. Moreover, the flux near the surface is diffusion dominated, so the loss of accuracy is not significant. Numerical results in Section 3.2 show that this scheme has much better accuracy than the standard Cartesian template.

2.6. Surface concentration

The boundary conditions at the reactive surfaces are implemented by balancing the diffusion of reactant towards the surface with the consumption of reactant by the dissolution kinetics, as indicated in Eq. (4). Since the solid–fluid boundary is not generally coincident with a grid point, we construct a virtual node (A) at a distance d from each surface marker, in the direction normal to the solid–fluid surface; the basic idea is illustrated in Fig. 8. The boundary condition, Eq. (4), can then be approximated by a first-order finite-difference:

$$D\frac{c_A - c_0}{d} = R(c_0), \tag{20}$$

where c_0 is the (unknown) concentration at the surface marker. Given the concentration at A and the kinetic rate law, $R(c_0)$, we can solve for the surface concentration, c_0 and the erosion flux $R(c_0)$. For linear kinetics, Eq. (5):

$$c_0 = \frac{dkc_{\mathrm{sat}} + Dc_A}{dk + D}, \tag{21}$$

but in general c_0 must be determined from an iterative solution of Eq. (20).

The value of the concentration at A is found by interpolation, using the surface normal at the marker point (Section 2.2) to place the point A at a fixed distance d from the surface; typically d is comparable to the grid spacing used in the concentration solver. If the point A lies inside a tetrahedron with four fluid vertexes, trilinear interpolation is used to determine the concentration at A. Otherwise, the concentration is taken from the nearest fluid point. This scheme avoids instabilities when a surface marker is very close to a fluid grid point and leads to robust and accurate predictions of the surface concentration (see Section 3.2). Using the surface normal and concentration, the erosion velocity, Eq. (6) can be determined, and the surface markers updated (Section 2.3).

2.7. Code parallelization

Modeling erosion or deposition in fractures and other porous media requires three different code modules: flow solver, concentration solver, and geometry update. We have implemented a three-dimensional domain decomposition for each of the modules independently. For the flow solver, information from the neighboring layer of the coarse grid must be passed to the adjacent processor. In Fig. 9, LB population densities from layer L1 in processor P1 are passed to processor P2. Similarly the concentrations in layer L2 must be passed to P2 as well. However, in some instances two layers of concentrations must be

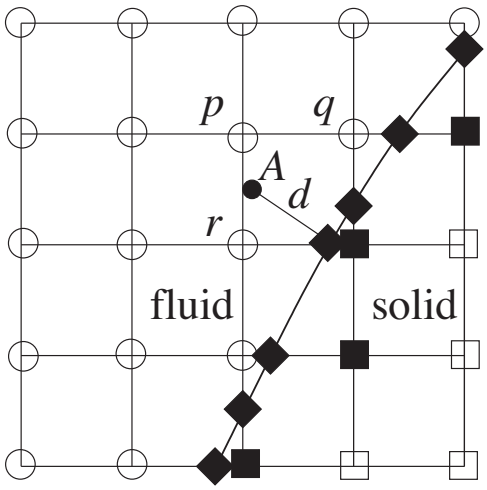

Fig. 8. Calculation of surface concentrations. The concentration at the surface markers (diamonds) is obtained from a finite-difference approximation, Eq. (20), to the boundary condition, Eq. (4). The concentration at A is obtained by interpolation.

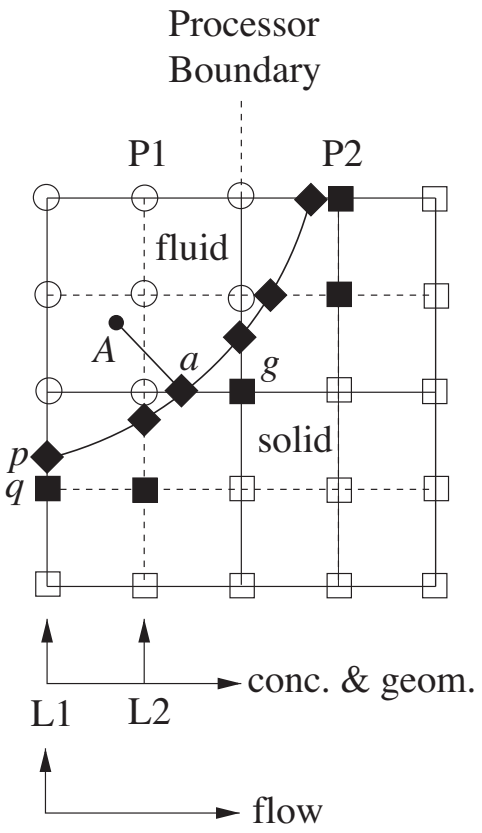

Fig. 9. Example of domain decomposition in the flow and concentration solvers. The coarse mesh of the flow solver is shown by solid lines and the finer grid of the concentration solver is shown by dashed lines.

passed. The solid boundary node g in the center of Fig. 9 belongs to both processors and thus the marker point a must be updated in both domains. Determining the concentration at the point A may require concentrations from the layer L1 as well as L2 so both must be passed.

In addition to the flow and concentration fields, geometric information must be communicated as well. This includes the location of the solid boundary nodes on the grid, the position of the surface markers with respect to the solid boundary nodes, and the normals at the marker points. In some instances geometric information must be passed from L1 as well as from L2 (Fig. 9). For example, the surface normal at the point a is calculated in both P1 and P2; the marker point p is therefore needed in both domains, since the nearer marker is too close for an accurate calculation of the normal (see Fig. 4, $\delta > 0.5$). Thus, boundary node data structures from L1, node q in this case, must be communicated as well as those from L2.

3. Numerical results

In this section we present numerical tests of the individual components; flow solver, concentration solver and geometry update. Finally the convergence of the whole simulation is examined in a complex, time-dependent topography.

3.1. Permeability of a random array of spheres

The convergence of the flow solver has been tested in a fixed but complex geometry, presented by a dense random pack of spheres. We have calculated the permeability of a single configuration of 108 identical spheres in a periodic unit cell of length $L = 10a$, where a is the particle radius. We used four different resolutions with radii $a = 2\Delta x$, $4\Delta x$, $8\Delta x$, and $16\Delta x$. An interpolated boundary condition [42] was used to obtain a second-order flow field throughout the domain.

The permeability is computed from the steady-state volumetric flow rate Q_x, driven by a uniform body force density $f_x \equiv -\partial_x p$:

$$\kappa = \frac{Q_x \eta}{A f_x}, \tag{22}$$

where η is the fluid viscosity and $A = L^2$ is the cross sectional area. Two different methods were used to calculate volumetric flow rate. In the simpler method, the flux in the x-direction is calculated by summing the velocity over all the grid points in each yz plane:

$$Q_x(x) = \sum_{y,z} v_x(x,y,z)\Delta x^2. \tag{23}$$

The overall permeability is obtained from the average flow rate over the whole volume.

The total volumetric flux through each yz plane should be the same in an incompressible flow, but in an irregular geometry the effective cross-sectional area associated with each grid point must be taken into account. A two-dimensional example can be used to indicate a more accurate calculation of the flux in a single volume element. In Fig. 10, the cell is cut by a solid body in the lower-right corner. The fluid portion of the cell is then divided into three triangles. In each triangle, the areal flux is calculated using following equation:

$$q_\triangle = \frac{v_1 + v_2 + v_3}{3}\frac{S_\triangle}{\Delta x}, \tag{24}$$

where v_1, v_2, and v_3 are the velocities at the three vertexes of a triangle and $S_\triangle$ is the area of the triangle. In three dimensions, the triangles are replaced by tetrahedra, and the surface area $S_\triangle$ in Eq. (24) is replaced by the volume of the tetrahedron. The total volumetric flux $Q_x(x + \Delta x/2)$ is obtained by summing over all the tetrahedra in cells lying between the planes x and $x + \Delta x$. The marching cube scheme [34] was used to construct the solid surface and the volume within each cell was then partitioned into tetrahedra. It should be noted that the configuration is not uniquely defined by a given set of marker points.

The variations in $Q_x(x)$ are shown in Fig. 11a for moderately sized spheres, $a = 8\Delta x$. In the first case (M1), Eq. (23), the effective cross-sectional area associated with each grid point has been ignored and the fluctuations in Q_x are large, but when

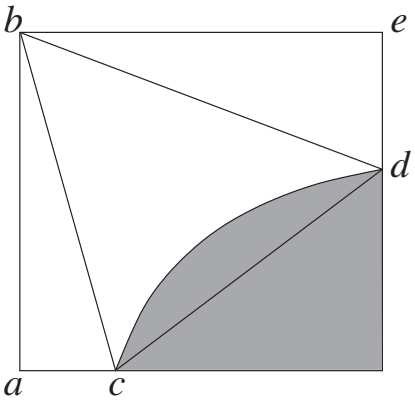

Fig. 10. Triangulation of a partially filled fluid cell.

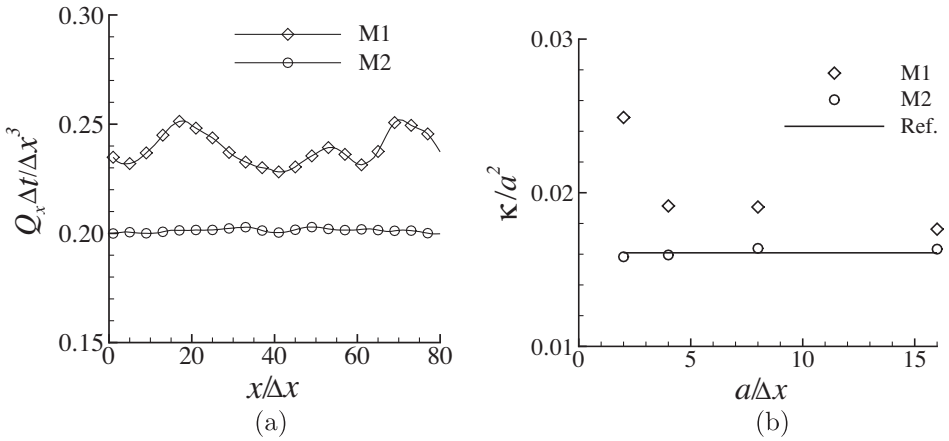

Fig. 11. The permeability of a dense pack of spheres in a cubic unit cell. In the left panel the flux, $Q_x(x)$, is shown as a function of position for spheres of radius $a = 8\Delta x$. In the right panel the average flux, $Q_x \propto \sum_x Q_x(x)$, is used to determine the permeability κ, Eq. (22). The solid line is the reference calculation, which uses a multipole expansion of the Green's function for Stokes flow [44].

the flow is summed over the individual tetrahedra (M2) the flux is much smoother. Moreover, the permeability obtained from the average Q_x is then almost independent of grid resolution, even for the smallest spheres, $a = 2\Delta x$ (Fig. 11b). The permeability agrees to within 1.5% with a highly accurate multipole solution of the same problem, derived from the methods described in Ref. [44]; a large number of multipoles, up to order 15, were used to ensure convergence of the final result. The small discrepancy between the multipole and LB results can be explained by the tetrahedral decomposition of the fluid volume, which replaces the curved surface of the spheres by planes. These inevitably cut into the convex surface of the sphere (Fig. 10), and slightly increase the fluid volume. On the other hand, a direct average of the velocity over the grid points (M1) produces an inaccurate permeability, even though the underlying flow field is the same as in M2. Large spheres $a > 10\Delta x$ are needed to get the permeability within 10%. These results emphasize the importance of accounting for local variations in porosity when calculating permeabilities from lattice-Boltzmann or other grid-based methods.

3.2. Concentration field in an angled channel

The concentration solver proposed in Sections 2.5 and 2.6 was tested by numerical simulations of a channel flow with chemically reacting boundaries. To increase the complexity of the problem, the channel was sometimes angled with respect to the underlying computational grid, as illustrated in Fig. 12. The width of the channel, $H = 10.02\Delta x$, was deliberately chosen to be a non-integer multiple of the grid spacing, so as to sample a range of distances between marker points and their solid boundary nodes, $0 < \delta < \Delta x$ (Fig. 1).

A linear kinetic model, Eq. (5), was used to describe the dissolution at the boundaries, with a rate constant adjusted so that the Damköhler number, $Da = k/\bar{u} = 0.08$, where $\bar{u}$ is the mean fluid velocity. The flow was driven by a uniform force density and characterized by a Péclet number $Pe = \bar{u}H/D = 125$, where H is the width of the channel; periodic boundary conditions were applied in the flow direction. The concentration field was generated by a circular source:

$$\frac{dc}{dt} = \begin{cases} 2.5 \times 10^{-4}(1-r)c_{\mathrm{sat}}/\Delta t, & r \leqslant 0.3H \\ 0.0, & r > 0.3H, \end{cases} \tag{25}$$

placed in the center of the channel.

The concentration flux at a marker point, Eq. (20), is calculated from a finite-difference with the concentration at a fixed distance d along the outward normal from the marker (see Section 2.6 and Fig. 8). A comparison of the surface concentration with a highly resolved solution from the NAG library [30] is shown in Fig. 13a; in this case the channel was aligned with the grid. The distance l along the boundary is measured from the projection of the center of the source onto the boundary surface. The surface concentrations for $d = \Delta x$ and $d = 0.5\Delta x$ are very similar and show a small 2% deviation from the reference solution, which is quite acceptable given the relatively coarse resolution ($h \approx 10\Delta x$). However, when the distance is too small, $d = 0.1\Delta x$, there are larger errors in the concentration profile. In all the remaining calculations $d = \Delta x$.

In an angled channel (slope = 0.5), we found that the standard template (D1), Eq. (17), leads to large errors in the concentration field, Fig. 13b. Moreover, there is an entirely unphysical asymmetry between the concentration at the upper and lower surfaces. We traced the error to the calculation of the convective flux and then replaced the standard five-point template with a one-dimensional flux calculated directly along the direction of the local velocity, Eq. (18) (centered) or Eq. (19) (upwind). The code makes the transition from centered to upwind differencing at a local Péclet number $Pe = 30$. Fig. 13b shows that the new template, illustrated in Fig. 7, gives a much more accurate solution, again within about 2% of the

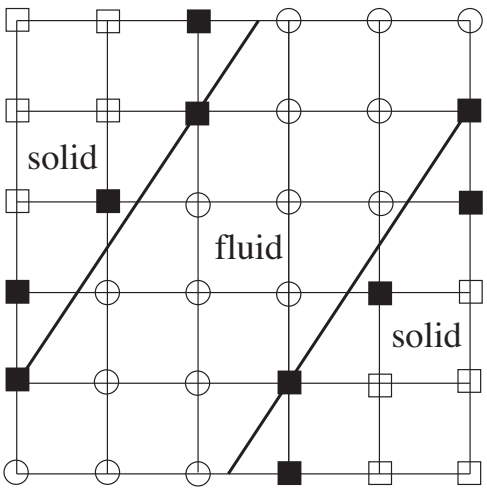

Fig. 12. Geometry of the angled channel.

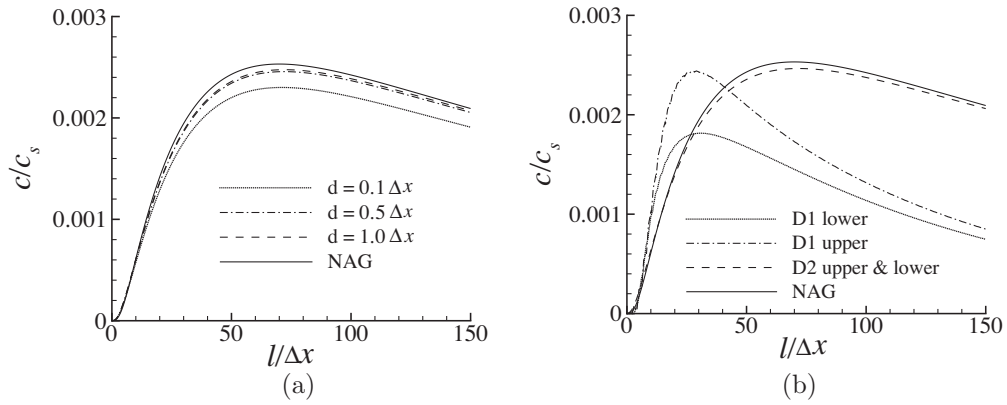

Fig. 13. Surface concentrations in a channel flow compared with a reference solution [30]; the distance l is measured in the flow direction from the center of the source. (a) Different values of the distance, d (Fig. 8). (b) Comparison of a standard finite-difference template (D1), Eq. (17), with differencing along the local flow direction, Eqs. (18) and (19) (D2).

reference calculation, and is symmetric at the upper and lower surfaces. This local differencing scheme seems to eliminate the artifacts associated with the calculation of a convective flux on a square grid and is useful in complex geometries such as porous media.

3.3. Testing the geometry update

The algorithm for moving the surface markers has been tested by simulating the uniform contraction of a sphere of radius $a = 8.5\Delta x$. The initial marker positions were obtained from the intersections of the sphere surface with the links. The sphere was then contracted at a constant velocity, so its exact radius was known at every time step. The position of each marker was updated by a displacement along its normal direction $-v\delta t$, where v is the speed of contraction. The surface normals were computed using the algorithm described in Section 2.2. In Fig. 14 the L_2 norm of the error in the surface position:

$$L_2 = \sqrt{\frac{\sum_m (a - r_m)^2}{Na^2}}, \tag{26}$$

is shown for two different velocities, $v = 0.01\Delta x/\Delta t$ and $v = 0.1\Delta x/\Delta t$; here N is the total number of markers, a is the (time-dependent) radius of sphere, and r_m is the distance of the marker from the origin.

The relative error in the marker position is roughly proportional to the square of the curvature, $C = \Delta x/a$, as shown in Fig. 14. At the final stage, $a = 2.5\Delta x$ the curvature is high, $C = 0.4$, yet the relative error in the marker position is only 2%. In addition to the curvature-dependent error, motion of the grid introduces an additional error that depends on the rate of contraction. These fluctuations are roughly proportional to velocity, but are not cumulative. Thus if a certain degree of raggedness in the short-time motion of the surface is acceptable, it can be moved at quite large velocities, of the order of $0.1\Delta x/\Delta t$.

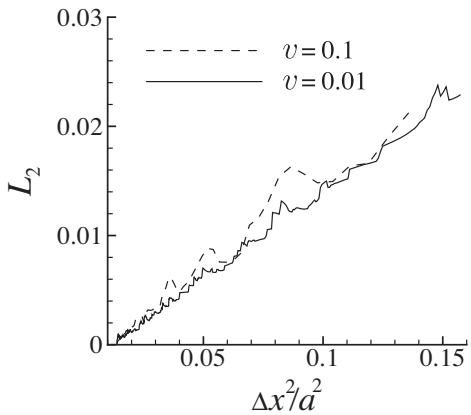

Fig. 14. The L_2 norm or the error in the marker positions on a moving spherical surface.

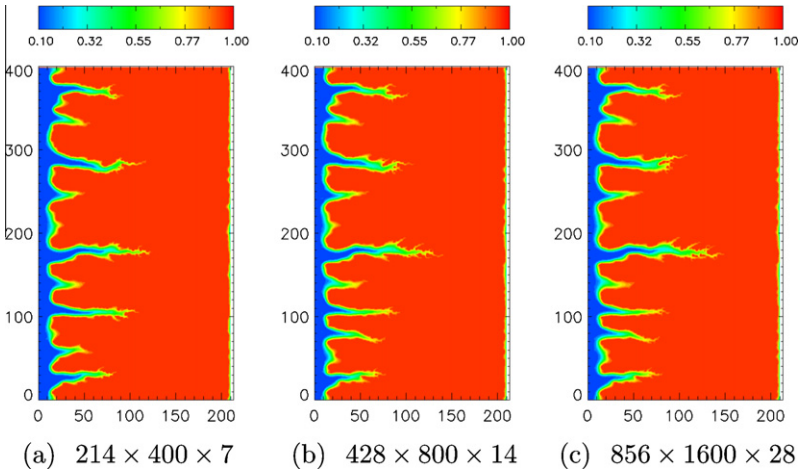

Fig. 15. Depth-averaged concentration fields at three different resolutions for dissolution at $Pe = 10$, $Da = 10$.

3.4. Convergence tests for dissolution simulations

The convergence of the entire algorithm was investigated by simulating dissolution inside an artificial fracture, made up of a narrow channel $200 \times 400 \times 3$ with small (3×3) obstacles randomly placed to block the flow [23]. The porosity of the fracture was approximately 50%. The computational domain was constructed by adding two more layers of solid above and below the fracture, to allow for dissolution, and insoluble inlet and outlet manifolds of length 7 units at the ends of the fracture. The geometry was mapped onto the computational grid using three different resolutions, $\Delta x = 1$ ($214 \times 400 \times 7$), $\Delta x = 2$ ($428 \times 800 \times 14$), and $\Delta x = 4$ ($856 \times 1600 \times 28$). The flow and concentration solvers used the same resolution in these tests.

A uniform force density (constant pressure gradient) was used to generate a flow field with an initial Péclet number $Pe = \bar{u}\bar{h}_0/D = 10$, and the reaction rate was chosen so that the initial Damköhler number $Da = k/\bar{u} = 10$; here $\bar{h}_0 = 1.5$ is the mean aperture of the initial fracture. The fracture is filled with a saturated solution, $c = 1$, and undersaturated fluid, $c = 0.1$, is released along a yz plane in the center of the inlet manifold, $x = 3.5$. Periodic boundary conditions on the flow and concentration fields were applied in the flow (x) direction. The small amount of erosion at the outlet is due to diffusion of concentration across the boundary, counter to the flow.

Fig. 15 shows the depth-averaged concentration field:

$$c_{av}(x,y) = \frac{1}{h(x,y)} \int c(x,y,z)dz, \tag{27}$$

at the time where the increase in mean aperture $\Delta\bar{h} = 0.5\bar{h}_0$. The erosion patterns are qualitatively similar in all three cases, despite the very coarse resolution in case (a), where there are only three grid points across the channel. A more detailed examination reveals that there are noticeable differences between the concentration fields in (a) and (b). The leading channel is somewhat shorter in the coarser simulation and the active channel at $y \sim 60$ in (a) has been replaced by an active channel at $y = 80$ in (b). The concentration fields in (b) and (c) are essentially identical. Thus although there are small differences in the three-dimensional fields, the key geophysical features of fracture dissolution are well represented with only $\sim$5 grid points across the channel.

4. Conclusions

In this paper a hybrid lattice-Boltzmann/finite-difference scheme for dissolution in fractured and porous media has been described. Bézier polynomials were used to construct a piecewise C^1 continuous surface, which is robust in complex topographies. A standard MRT lattice-Boltzmann method with interpolated boundary conditions was used to solve for the flow field at each iteration. However, in the absence of an upwind differencing scheme, LB methods are limited in the range of grid Péclet numbers that can be accessed, and we therefore used a finite-difference method for the concentration field. Nevertheless, we found it useful to borrow the idea of advection along characteristics from the LB methodology; this allowed for a much more accurate convective flux in complex topographies when the flow field does not follow the grid lines.

Numerical results were presented to validate the numerical algorithms. It was shown that the flow solver leads to accurate permeabilities if the local porosity is properly accounted for. We found that when the flow has a substantial normal component towards a nearby surface, the standard finite-difference templates for the convective flux lead to highly inaccurate results. An alternative differencing scheme, based on the local flow velocity, was shown to lead to good agreement with a reference calculation. A simple numerical example was used to demonstrate the fidelity of the method we used to move

the marker points. Finally we showed that there is good overall convergence of the erosion patterns in a complex porous topography.

The algorithms presented in this paper can be used to simulate dissolution in laboratory scale samples, where the sample dimensions are typically of the order of 10×10 cm, with a mean aperture of approximately 0.1 mm [45]. Based on the results presented in Section 3.4, we would expect satisfactory dissolution patterns with about four grid points across the aperture in the flow solver and eight in the concentration solver. The sample volume would then contain about 10^7 LB grid points and about 10^8 grid points for the concentration solver. Such systems easily fit within the memory of a small cluster, including the memory required to store the surface information. In fact we have previously made such simulations on a single workstation, although using a stochastic simulation for the reactant transport [21]. Extensions to the field scale, say 1×1 m, should be possible on a supercomputer (>10^4 processors).

Acknowledgments

This work was supported by the US Department of Energy, Chemical Sciences, Geosciences and Biosciences Division, Office of Basic Energy Sciences (DE-FG02–98ER14853). We thank Elek Wajnryb (Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland) for providing the multipole code used to calculate the reference solution in Section 3.1.

References

[1] K.T.B. MacQuarrie, K.U. Mayer, Reactive transport modeling in fractured rock: a state-of-the-science review, Earth Sci. Rev. 72 (2005) 189–227.
[2] C.I. Steefel, Geochemical kinetics and transport, in: S.L. Brantley, J.D. Kubicki, A.F. White (Eds.), Kinetics of Water-Rock Interaction, Springer, New York, 2007, pp. 545–589.
[3] B. Cailly, P.L. Thiez, P. Egermann, A. Audibert, S. Vidal-Gilbert, X. Longaygue, Geological storage of CO_2: a state-of-the-art of injection processes and technologies, Oil Gas Sci. Technol. 60 (2005) 517–525.
[4] J. Ennis-King, L. Paterson, Coupling of geochemical reactions and convective mixing in the long-term geological storage of carbon dioxide, Int. J. Green Gas Cont. 1 (2007) 86–93.
[5] M.L. Hoefner, H.S. Fogler, Pore evolution and channel formation during flow and reaction in porous media, AIChE J. 34 (1988) 45–54.
[6] J. Siemers, W. Dreybrodt, Early development of karst aquifers on percolation networks of fractures in limestone, Water Resour. Res. 34 (1998) 409–419.
[7] R.B. Hanna, H. Rajaram, Influence of aperture variability on dissolutional growth of fissures in karst formations, Water Resour. Res. 34 (1998) 2843–2853.
[8] F. Golfier, C. Zarcone, B. Bazin, R. Lenormand, D. Lasseux, M. Quintard, On the ability of a Darcy-scale model to capture wormhole formation during the dissolution of a porous medium, J. Fluid Mech. 457 (2002) 213–254.
[9] C. Cohen, D. Ding, M. Quintard, B. Bazin, From pore scale to wellbore scale: impact of geometry on wormhole growth in carbonate acidization, Chem. Eng. Sci. 63 (2008) 3088–3099.
[10] W. Cheung, H. Rajaram, Dissolution finger growth in variable aperture fractures: role of the tip-region flow field, Geophys. Res. Lett. 29 (2002) 2075.
[11] S. Békri, J.-F. Thovert, P.M. Adler, Dissolution of porous media, Chem. Eng. Sci. 50 (1995) 2765–2791.
[12] S. Békri, J.-F. Thovert, P.M. Adler, Dissolution and deposition in fractures, Eng. Geol. 48 (1997) 283–308.
[13] R. Verberg, A.J.C. Ladd, Simulation of chemical erosion in rough fractures, Phys. Rev. E 65 (2002) 056311.
[14] Q.J. Kang, D.X. Zhang, S.Y. Chen, Simulation of dissolution and precipitation in porous media, J. Geophys. Res. 108 (2003) B102505.
[15] Q.J. Kang, P.C. Lichtner, D.X. Zhang, An improved lattice Boltzmann model for multicomponent reactive transport in porous media at the pore scale, Water Resour. Res. 43 (2007) W12S14.
[16] Q.J. Kang, P.C. Lichtner, H.S. Viswanathan, A.I. Abdel-Fattah, Pore scale modeling of reactive transport involved in geologic CO_2 sequestration, Transport Porous Med. 82 (2010) 197–213.
[17] R. Verberg, A.J.C. Ladd, Lattice-Boltzmann model with sub-grid scale boundary conditions, Phys. Rev. Lett 84 (2000) 2148–2151.
[18] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, L.S. Luo, Multiple-relaxation-time lattice Boltzmann models in three dimensions, Philos. Trans. R. Soc. Lond. A 360 (2002) 437–451.
[19] P. Szymczak, A.J.C. Ladd, Boundary conditions for stochastic solutions of the convection–diffusion equation, Phys. Rev. E. 68 (2003) 036704.
[20] P. Szymczak, A.J.C. Ladd, Stochastic boundary conditions to the convection–diffusion equation including chemical reactions at solid surfaces, Phys. Rev. E. 69 (2004) 036704.
[21] P. Szymczak, A.J.C. Ladd, Microscopic simulations of fracture dissolution, Geophys. Res. Lett. 31 (2004) L23606.
[22] P. Szymczak, A.J.C. Ladd, A network model of channel competition in fracture dissolution, Geophys. Res. Lett. 33 (2006) L05401.
[23] P. Szymczak, A.J.C. Ladd, Wormhole formation in dissolving fractures, J. Geophys. Res. 114 (2009) B06203.
[24] P.B. Warren, Electroviscous transport problems via lattice-Boltzmann, Int. J. Mod. Phys. C 8 (1997) 889–898.
[25] I. Rasin, S. Succi, W. Miller, A multi-relaxation lattice kinetic method for passive scalar diffusion, J. Comp. Phys. 205 (2005) 451–462.
[26] I. Ginzburg, Equilibrium-type and link-type lattice Boltzmann models for generic advection and anisotropic-dispersion equation, Adv. Water Resour. 28 (2005) 1171–1195.
[27] J.A. Kaandorp, C.P. Lowe, D. Frenkel, P.M.A. Sloot, Effect of nutrient diffusion and flow on coral morphology, Phys. Rev. Lett. 77 (1996) 2328–2331.
[28] D. Yu, S.S. Girimaji, A.J.C. Ladd, Revised moment propagation method for scalar transport, Phys. Rev. E 78 (2008) 056706.
[29] M. Stiebler, J. Tölke, M. Krafczyk, Advection-diffusion lattice Boltzmann scheme for hierarchical grids, Comput. Math. Appl. 55 (2008) 1576–1584 (Mesoscopic Methods in Engineering and Science).
[30] NAG, NAG Fortran Library Manual, Mark 18, The Numerical Algorithms Group Limited, Oxford, 1997.
[31] S. Chen, G.D. Doolen, Lattice Boltzmann method for fluid flows, in: J.L. Lumley, M.V. Dyke, H.L. Reed (Eds.), Annual Review of Fluid Mechanics, vol. 30, Annual Reviews Inc, Palo Alto, California, 1998, pp. 329–364.
[32] C. Montani, R. Scateni, A modified look-up table for implicit disambiguation of marching cubes, Visual Comput. 10 (1994) 353–355.
[33] T.N.T. Goodman, H.B. Said, A C^1 triangular interpolant suitable for scattered data interpolation, Commun. Appl. Numer. M 7 (1991) 479–485.
[34] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, Comput. Graph. 21 (1987) 163–169.
[35] B. Ahrenholz, J. Tölke, M. Krafczyk, Lattice-Boltzmann simulations in reconstructed parametrized porous media, Int. J. Comput. Fluid D 20 (2006) 369–377.
[36] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, J.-P. Rivet, Lattice gas hydrodynamics in two and three dimensions, Complex Syst. 1 (1987) 649.
[37] Y.H. Qian, D. d'Humières, P. Lallemand, Lattice BGK models for the Navier–Stokes equation, Europhys. Lett. 17 (1992) 479–484.
[38] D. d'Humières, Generalized lattice Boltzmann equations, Prog. Astronaut. Aeronaut. 159 (1992) 450–458.

[39] B. Chun, A.J.C. Ladd, Interpolated boundary condition for lattice-Boltzmann simulations of flows in narrow gaps, Phys. Rev. E 75 (2007) 066705.
[40] I. Ginzburg, D. d'Humières, Multireflection boundary conditions for lattice Boltzmann models, Phys. Rev. E 68 (2003) 066614.
[41] M. Bouzidi, M. Firdaouss, P. Lallemand, Momentum transfer of a Boltzmann-lattice fluid with boundaries, Phys. Fluids 13 (2001) 3452–3459.
[42] D.Z. Yu, R.W. Mei, L.S. Luo, W. Shyy, Viscous flow computations with the method of lattice Boltzmann equation, Prog. Aerosp. Sci 39 (2003) 329–367.
[43] W. Shyy, Computational Modeling for Fluid Flow and Interfacial Transport, Elsevier, Amsterdam, The Netherlands, 1994.
[44] B. Cichocki, R.B. Jones, R. Kutteh, E. Wajnryb, Friction and mobility for colloidal spheres in Stokes flow near a boundary: the multipole method and applications, J. Chem. Phys. 112 (2000) 2548–2561.
[45] R.L. Detwiler, R.J. Glass, W.L. Bourcier, Experimental observations of fracture dissolution: the role of Péclet number in evolving aperture variability, Geophys. Res. Lett. 30 (2003) 1648.