Algorithm to implement unsteady jump boundary conditions within the lattice Boltzmann method

Badr Kaoui

EPJ manuscript No.
(will be inserted by the editor)

Algorithm to implement unsteady jump boundary conditions within the lattice Boltzmann method

Badr Kaoui[1a]

Biomechanics and Bioengineering Laboratory,
Université de Technologie de Compiègne, CNRS, 60200 Compiègne, France

Received: date / Revised version: date

Abstract. An algorithm is proposed to implement unsteady jump boundary conditions in the lattice Boltzmann method (LBM). This is useful for dealing with problems of mass transfer across membranes that exhibit resistance and discontinuity in concentration. The algorithm is simple to implement into an existing LBM-based code that computes diffusion and advection of a solute. Analytical solutions are recovered in the limiting case of a planar membrane. When combined with the immersed boundary method, the algorithm can handle moving deformable boundaries that adopt arbitrary geometries. Simulations of controlled solute release from stationary rigid and moving deformable particles are given as a proof of concept.

PACS. PACS-key discribing text of that key – PACS-key discribing text of that key

1 Problem statement

Transport of chemical species across interfaces or membranes is encountered in many natural phenomena and technological processes. For example, most of the material exchange in biological and biomedical systems take place across membranes [1,2]. The membranes act as a barrier to filter and control the rate of material exchange between two adjacently separated environments. The main property that characterizes this transport functionality, at the macroscopic scale, is the permeability or its inverse the resistance.

Modeling the effect of a restricted permeability that leads to a jump in the concentration at the membrane is delicate. The resulting discontinuity represents numerical challenges, specially when the membrane is moving and adopting irregular geometrical configurations. Few works have recently been proposed to handle this issue, with applications, ranging from stationary planar interfaces or membranes to moving deformable particles. Membranes with restrictive permeability to both, solute and solvent, or only to one of them have been considered. For example, Wang [3] and Hickson *et al.* [4] have proposed finite difference schemes, Layton [5] and Jayathilake *et al.* [6,7] have used the immersed interface method [8], Miyauchi *et al.* [9, 10] have proposed a finite element discretization method, Huang *et al.* [11,12] have used the immersed boundary method (IBM) [13], and Yao *et al.* [14] have combined the IBM with a Cartesian grid embedded boundary method. In the present article, another alternative algorithm is proposed for the lattice Boltzmann method (LBM).

The LBM has emerged as a modern efficient numerical method for computational fluid dynamics (CFD) [15–19]. It is a discrete particle-based method that is capable to simulate properly the dynamics of complex fluids that flow in complex geometries [20]. It has been demonstrated that the LBM can recover the solution of many partial differential equations [15], such as the advection-diffusion equation that is the subject of this work. Different schemes have been proposed to implement boundary conditions for solute concentration in the LBM: (i) The bounce-back boundary conditions that consist in reflecting back the advected distribution populations that hit a solid boundary. A generalized version for mass transport has been proposed by Zhang *et al.* [21], (ii) Estimating the unknown distribution populations at the boundaries, such as proposed by Inamuro *et al.* [22] for temperature and which can be also used for solute concentration, (iii) Evaluating the unknown physical quantities at curved boundaries using finite difference method techniques, as done by Guo *et al.* [23], (iv) Using a forcing term combined with the immersed boundary method for moving objects, as proposed by Kasparek *et al.* [24]. In this work, the forcing term strategy is also used but to incorporate a time-varying jump at a zero-thickness membrane separating two domains, as illustrated in Fig. 1. The mathematical formulation of the problem is given in Sec. 2, followed by a brief presentation of the LBM solver in Sec. 3. The main steps of the algorithm are detailed in Sec. 4, and the parameter calibration explained in Sec. 5. Two applications of the algorithm, mass transfer across stationary and moving membranes are given in Sec. 6, before concluding the article by Sec. 7.

[a] badr.kaoui@utc.fr

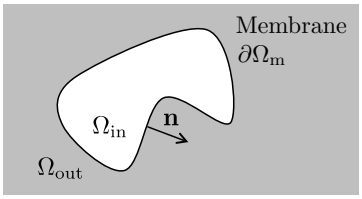

Fig. 1. The computational domain composed of two adjacent subdomains, Ω_{in} and Ω_{out}, separated by a closed zero-thickness membrane $\partial\Omega_m$. $\mathbf{n}$ is the unit normal vector to the membrane.

2 Mathematical formulation

The computational domain is composed of two adjacent subdomains Ω_{in} and Ω_{out}, which are separated by a zero-thickness membrane $\partial\Omega_m$ (see Fig. 1). Diffusion and advection of a solute in each subdomain are given by,

$$\frac{\partial c}{\partial t} + (\mathbf{v}\cdot\nabla c) = D_{in}\nabla^2 c \quad \text{if} \quad \mathbf{r}\in\Omega_{in}, \tag{1}$$

and

$$\frac{\partial c}{\partial t} + (\mathbf{v}\cdot\nabla c) = D_{out}\nabla^2 c \quad \text{if} \quad \mathbf{r}\in\Omega_{out}. \tag{2}$$

$\mathbf{r}$ is the vector position. $c(\mathbf{r},t)$ and $\mathbf{v}(\mathbf{r},t)$ are respectively the instantaneous local solute concentration and the solvent velocity. D_{in} and D_{out} are the diffusion coefficients of the solute in the inner and the outer domains, respectively. The initial condition of interest for this study is,

$$c(\mathbf{r},t=0) = \begin{cases} 1 & \text{if } \mathbf{r}\in\Omega_{in} \\ 0 & \text{if } \mathbf{r}\in\Omega_{out}, \end{cases} \tag{3}$$

that models an initially loaded particle. This condition establishes a concentration gradient, at the membrane, which triggers solute transport from Ω_{in} to Ω_{out} with a mass flux along the normal direction given by the jump boundary condition (see Fig. 2),

$$\mathbf{J}\cdot\mathbf{n} = P\,[c]_{\partial\Omega_m} = P\,(c_{in}-c_{out}) \quad \text{if} \quad \mathbf{r}\in\partial\Omega_m, \tag{4}$$

where $\mathbf{J}$ is the mass flux through the membrane, $\mathbf{n}$ is the unit normal vector on the membrane that points from Ω_{in} to Ω_{out}, and P the intrinsic permeability of the membrane. $[c]_{\partial\Omega_m} = (c_{in}-c_{out})$ is the jump discontinuity, with c_{in} and c_{out} are the concentrations on the inner and the outer sides of the membrane $\partial\Omega_{in}$ and $\partial\Omega_{out}$, respectively.

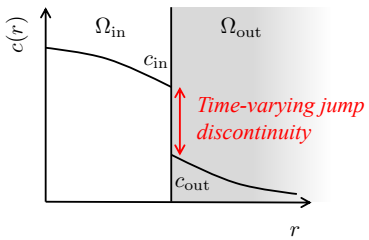

Fig. 2. Typical concentration profile $c(r)$ in the direction normal to a membrane, whose finite resistance leads to the jump discontinuity $[c]_{\partial\Omega_m} = (c_{in}-c_{out})$.

Solving Eqs. 1 and 2, while considering the boundary condition Eq. 4 is no easy task. A set of assumptions are considered in this study:

- Two-dimensional simulations are performed $\mathbf{r}\equiv(x,y)$,
- P is constant and uniform along the membrane,
- At equilibrium, when $t\to\infty$,

$$c_{out}(\mathbf{r},\infty) = c_{in}(\mathbf{r},\infty) = c_{eq} \quad \text{for} \quad \mathbf{r}\in\partial\Omega_m, \tag{5}$$

- The membrane is solvent-impermeable, which also implies the continuity of the normal component of the velocity across the membrane,
- One-way coupling is considered. That is the flow alters locally the solute concentration via the advection term in Eqs. 1 and 2, while the solute has no impact neither on the flow nor on the properties of the solvent,
- No-slip velocity is considered on the membrane, which implies the continuity of the tangential component of the velocity across the membrane,
- The influx and outflux across the membrane are equal:

$$\begin{aligned}[c_{in}\,(\mathbf{v}-\mathbf{v}_m) - D_{in}\nabla c|_{\partial\Omega_{in}}]\cdot\mathbf{n} = \\ [c_{out}\,(\mathbf{v}-\mathbf{v}_m) - D_{out}\,(\nabla c|_{\partial\Omega_{out}})]\cdot\mathbf{n} \\ \text{for} \quad \mathbf{r}\in\partial\Omega_m,\end{aligned} \tag{6}$$

  where $\nabla c|_{\partial\Omega_{in}}$ and $\nabla c|_{\partial\Omega_{out}}$ are the derivatives of the concentration on either side of the membrane, and $\mathbf{v}_m$ the membrane velocity. $\mathbf{v}_m = \mathbf{v}$ since the normal and tangential components of the velocity are assumed to be continuous across the membrane. Thus, the above equality is reduced to:

$$\begin{aligned}D_{in}\,(\nabla c|_{\partial\Omega_{in}}\cdot\mathbf{n}) = D_{out}\,(\nabla c|_{\partial\Omega_{out}}\cdot\mathbf{n}) \\ \text{for} \quad \mathbf{r}\in\partial\Omega_m,\end{aligned} \tag{7}$$

- The diffusion coefficient is similar in both subdomains:

$$D = D_{in} = D_{out}. \tag{8}$$

One-dimensional analytical solution under the above assumptions exists, as given by Crank [25],

$$\begin{aligned}c(x,t) = \frac{1}{2}\Bigg[1 + \Bigg\{&\operatorname{erf}\frac{x}{2\sqrt{Dt}} \\ &+ \exp\left(ax + a^2Dt\right)\operatorname{erfc}\left(\frac{x}{2\sqrt{Dt}} + a\sqrt{Dt}\right)\Bigg\}\Bigg],\end{aligned} \tag{9}$$

for $x>0$, and

$$\begin{aligned}c(x,t) = \frac{1}{2}\Bigg\{&\operatorname{erfc}\frac{|x|}{2\sqrt{Dt}} \\ &- \exp\left(a|x| + a^2Dt\right)\operatorname{erfc}\left(\frac{|x|}{2\sqrt{Dt}} + a\sqrt{Dt}\right)\Bigg\},\end{aligned} \tag{10}$$

for $x<0$ with

$$a = 2\frac{P}{D}. \tag{11}$$

P is used as a fitting parameter for calibration in Sec. 5.

3 Lattice Boltzmann method

The first necessary ingredient for the algorithm is a mass transfer solver. In the present work, the LBM is adopted instead of solving directly Eqs. 1 and 2 using, for example, the finite difference method. The LBM has emerged as an alternative fluid flow solver in the last decades with high potential to handle complex fluid dynamics, while avoiding the complicated weak formulations and the extensive pre-processing meshing needed by the finite element method. More details about LBM could be found in the following textbooks Refs. [15–19].

The main quantity of interest in the LBM is the distribution function g_i. For the D2Q9 lattice, used in this study, the two-dimensional discrete position is given by $\mathbf{r} \equiv (x, y)$ and the nine discrete velocity directions by $\mathbf{e}_i$, with $i = 0 \rightarrow 8$. The evolution in time of g_i is given by the lattice-Boltzmann equation,

$$g_i(\mathbf{r}+\mathbf{e}_i, t+1) - g_i(\mathbf{r}, t) = -\frac{g_i(\mathbf{r}, t) - g_i^{\mathrm{eq}}(\mathbf{r}, t)}{\tau_{\mathrm{g}}} + \omega_i S_i. \quad (12)$$

The first term on the right-hand side is the Bhatnagar-Gross-Krook operator [26], and it gives the relaxation of g_i towards its equilibrium distribution g_i^{eq} [27],

$$g_i^{\mathrm{eq}}(\mathbf{r}, t) = \omega_i\, c(\mathbf{r}, t) \left[1 + 3(\mathbf{v}\cdot\mathbf{e}_i) + \frac{9}{2}(\mathbf{v}\cdot\mathbf{e}_i)^2 - \frac{3}{2}(\mathbf{v})^2\right], \quad (13)$$

where ω_i are the D2Q9 lattice weight factors: $\omega_i = 4/9$ for $i = 0$, $\omega_i = 1/9$ for $i = 1, 2, 3, 4$ and $\omega_i = 1/36$ for $i = 5, 6, 7, 8$. The microscopic relaxation time τ_{g} in Eq. 12 is related to the macroscopic diffusion coefficient via $D = \frac{1}{3}\left(\tau_{\mathrm{g}} - \frac{1}{2}\right)$. The second term on the right-hand side of Eq. 12 expresses the forcing term with S_i is the source term. The zeroth order moment of g_i gives the local concentration of the solute

$$c(x, y, t) = \sum_{i=0}^{8} g_i(x, y, t). \quad (14)$$

All the parameters and the physical quantities in this work are given in the dimensionless lattice units. The spatial and the temporal discretization used in Eq. 12 and hereafter are all set to $h = \Delta x = \Delta y = \Delta t = 1$, which is the convention adopted in the LBM. A rigorous physically-based explanation of the lattice units is given by Succi [16], and a practical conversion between the lattice and the physical units (SI) is proposed by Dupin *et al.* [28].

4 Algorithm

The algorithm is proposed to handle study cases that involve unsteady mass transfer across moving membranes with zero thickness and having a finite permeability. The resulting jump in the concentration at the membrane is incorporated using a forcing term, somehow similar to the way of including hydrodynamic stress jump at moving boundaries in the immersed boundary method [13].

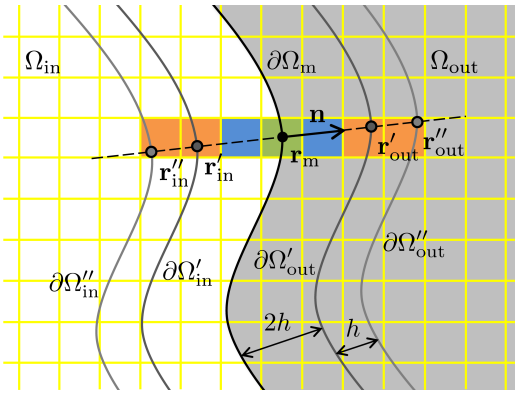

Fig. 3. The membrane $\partial\Omega_{\mathrm{m}}$, its four closer enveloping layers ($\partial\Omega'_{\mathrm{in}}$, $\partial\Omega''_{\mathrm{in}}$, $\partial\Omega'_{\mathrm{out}}$ and $\partial\Omega''_{\mathrm{out}}$), and the LBM regular grid.

The main steps of the algorithm can be implemented in any existing LBM-based code. Given the actual membrane position $\mathbf{r}_{\mathrm{m}}(t)$ and the concentration scalar field $c(x, y, t)$ in the whole computational domain Ω, the scalar forcing term $S_i(x, y, t)$ needed to compute the concentration $c(x, y, t+1)$ at the next time iteration is evaluated following these consecutive steps:

1. Localize the inner and the outer enveloping layers of the membrane,
$$\partial\Omega'_{\mathrm{in}}, \quad \partial\Omega''_{\mathrm{in}}, \quad \partial\Omega'_{\mathrm{out}} \quad \text{and} \quad \partial\Omega''_{\mathrm{out}}, \quad (15)$$
whose distances from the actual location of the membrane are $2h$ and $3h$, as illustrated in Fig. 3,
2. Compute the solute concentration along each layer using bilinear interpolation. The known concentrations on the four nodes of the orange colored elements in Fig. 3 give the concentration, for example, at $\mathbf{r}'_{\mathrm{out}}$ as:
$$c(\mathbf{r}'_{\mathrm{out}}, t) = \sum_{\{x,y\}} \alpha(x, y, \mathbf{r}'_{\mathrm{out}}) c(x, y, t). \quad (16)$$
where α is used as an interpolation weight [28]. It is the area shared by an off-lattice point $\mathbf{r}$ and one of its four neighboring enveloping on-lattice points $\{x, y\}$, as illustrated in Fig. 4. Layers that are $1h$ distant from the membrane, as is the case in the blue colored elements in Fig. 3, are discarded because there the concentration and its local derivatives are not necessarily

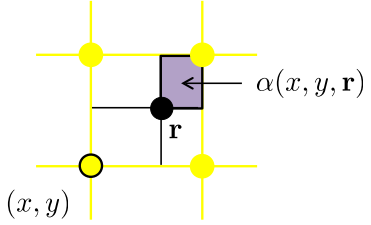

Fig. 4. An off-lattice point $\mathbf{r}$, and one of its four neighboring on-lattice points (x, y). The area $\alpha(x, y, \mathbf{r})$ is the interpolation weight associated to the concentration at (x, y) used to evaluate the concentration at $\mathbf{r}$, using Eq. 16.

continuous to perform interpolation (see Fig. 7). The smooth Diract function, largely used in the immersed boundary method [13], is not adapted here to preserve the sharp jump discontinuity. It smears out numerically the concentration,

3. Estimate the concentrations on either side of the membrane using linear extrapolation,

$$c_{\rm in}(\mathbf{r}_{\rm m},t) = 3c(\mathbf{r}'_{\rm in},t) - 2c(\mathbf{r}''_{\rm in},t), \tag{17}$$
$$c_{\rm out}(\mathbf{r}_{\rm m},t) = 3c(\mathbf{r}'_{\rm out},t) - 2c(\mathbf{r}''_{\rm out},t). \tag{18}$$

4. Evaluate the *ad hoc* source term $\mathbf{q}(\mathbf{r}_{\rm m},t)$ along the membrane by multiplying the estimated jump in the concentration by a numerical prefactor R, hereafter called resistance,

$$\mathbf{q}(\mathbf{r}_{\rm m},t) = -R\left[c_{\rm in}(\mathbf{r}_{\rm m},t) - c_{\rm out}(\mathbf{r}_{\rm m},t)\right]\mathbf{n}. \tag{19}$$

5. Spread the source term $\mathbf{q}(\mathbf{r}_{\rm m},t)$ throughout the overall computational domain using,

$$\mathbf{Q}(x,y,t) = \int_{\partial\Omega_{\rm m}} \alpha(x,y,\mathbf{r}_{\rm m})\mathbf{q}(\mathbf{r}_{\rm m},t){\rm d}s, \tag{20}$$

that has non-zero values solely on lattice nodes (x,y) that surround closely each membrane node $\mathbf{r}_{\rm m}$,

6. Plug the source term $\mathbf{Q}(x,y,t)$, after its projection into the velocity space, in the lattice Boltzmann equation as a forcing term :

$$g_i(\mathbf{r}+\mathbf{e}_i,t+1) - g_i(\mathbf{r},t) = -\frac{g_i(\mathbf{r},t) - g_i^{\rm eq}(\mathbf{r},t)}{\tau_{\rm g}} + \omega_i\left(\mathbf{Q}(x,y,t)\cdot\mathbf{e}_i\right). \tag{21}$$

In this way, the term $S_i = \mathbf{Q}(x,y,t)\cdot\mathbf{e}_i$ allows to control the mass flux rate across the membrane. It acts as a source term on the inner side of the membrane, and as a sink term on the outer side of the membrane. The projection of $\mathbf{Q}$ into the velocity space gives the possibility to set either a sink or a source term on each velocity direction of a giving lattice node. The scalar product $\mathbf{n}\cdot\mathbf{e}_i$ assigns either a positive or a negative sign to the final forcing term plugged into the lattice Boltzmann equation. Thus, the source term is velocity direction dependent, here, in contrast to the classical way of incorporating source terms into the LBM. For a planar membrane parallel to the y axis, at each of its nodes, the above projection attributes source terms on the directions $\mathbf{e}_5$, $\mathbf{e}_1$ and $\mathbf{e}_8$, and sink terms on $\mathbf{e}_6$, $\mathbf{e}_3$ and $\mathbf{e}_7$. By the way, the projection conserves mass.

The evolution in time of the concentration on the inner side of the membrane is given by the balance equation,

$$V\frac{{\rm d}c_{\rm in}}{{\rm d}t} = -AP(c_{\rm in} - c_{\rm out}), \tag{22}$$

where V is the volume and A the area of the donor domain, here the inner domain $\Omega_{\rm in}$. The concentration $c_{\rm in}$ drops down, and this leads to an increase of the concentration on the outer side $c_{\rm out}$. The algorithm recovers the same mechanism; however, by imposing a higher concentration on the inner side of the membrane and a lower value on the outer side, both with respect to an average value. With this alternative way, the algorithm is capable to recover approximately the solutions of Eqs. 1 and 2, with the initial condition Eq. 3 and the jump boundary condition Eq. 4.

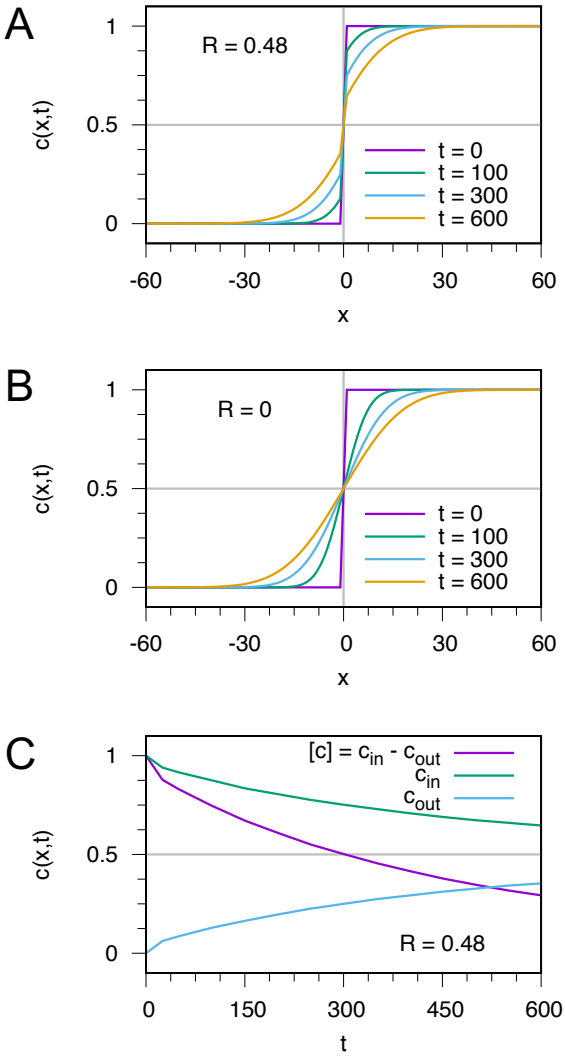

Fig. 5. Evolution in time of numerically computed concentration profiles $c(x,t)$, in the x-direction, for a planar membrane that has a finite resistance $R = 0.48$ (a). For comparison purpose, profiles obtained for $R = 0$ are also shown in (b). (c) Temporal evolution of the concentrations on either side of the membrane, $c_{\rm in}$ and $c_{\rm out}$, and the resulting decay of the jump discontinuity $[c]_{\partial\Omega_{\rm m}} = (c_{\rm in} - c_{\rm out})$.

Figure 5a shows the obtained concentration profiles over time for a planar membrane when using the algorithm with an arbitrary non-zero value of the resistance $R = 0.48$. The profiles are steep. They demonstrate slow diffusion process compared to the case with zero resistance $R = 0$, shown in Fig. 5b, which corresponds to a membrane with infinite permeability ($P \to \infty$). The algorithm produces successfully the decay over time of the jump discontinuity at the membrane, as quantified in Fig. 5c. Details of these simulations are given in the next Sec. 5.

5 Calibration and convergence analysis

The non-trivial step in the algorithm is how to set an adequate value for the resistance R (Step 4 in Sec. 4), which is capable to reproduce the same mass transport scenario as a membrane with a given permeability P does. This issue could be solved by running a series of trial and error tests to calibrate the computer code. The one-dimensional analytical solution, given by Eqs. 9 and 10, is used as a reference. During each calibration test, a value is attributed to R and the computed concentration profiles are fit with the analytical solution by fine-tuning P until accomplishing acceptable fit. Two-dimensional simulations are performed on a square domain of size 200×200, where a straight line representing a planar membrane is placed parallel to the y-axis and at the origin of the x-axis ($x_\mathrm{m} = 0$). The membrane is discretized with 199 nodes uniformly separated by $h = 1$. Zero-flux is set on both domain edges in the x-direction using the bounce-back boundary condition [17]. Periodic boundary condition is set along the y-direction in order to model an infinite direction; and thus to recover the one-dimensional solution. The initial concentration is set to unity for $x > 0$, to zero for $x < 0$ and to half for $x = 0$.

Computed concentration profiles at different elapsed times t for a non-zero value of the resistance $R = 0.48$, with their respective fit with $P = 0.012$, are reported in Figs. 6a. The numerical data are represented by symbol points, while the analytical solution with solid lines. The Fig. 6a shows a zoom into the x range $[-30, 30]$. The fit is not excellent all the time since a single value of P could not achieve perfect data fit at all times. There is a range of P, here within the interval $[0.01, 0.02]$, that may fit well the obtained data but with different quality. This quality is measured by the instantaneous root-mean-square error

$$\mathrm{RMSE}(t) = \frac{1}{n_x} \sum_{x=0}^{n_x} \left(c_\mathrm{num}(x,t) - c_\mathrm{theo}(x,t) \right)^2 , \qquad (23)$$

where n_x is the half length of the computational domain along the x-direction (number of the grid points with $x > 0$), $c_\mathrm{num}(x,t)$ is the computed solution and $c_\mathrm{theo}(x,t)$ the theory (Eqs. 9 and 10). Because the concentration profiles are antisymmetric with respect to the point ($x = 0$, $c = 0.5$), only the error for $x > 0$ is reported in Fig. 6b. The perfect antisymmetric character of the curves is also a signature of mass conservation. What is lost on one side of the membrane is totally recovered by the other side. RMSE(t) measures the deviation of the numerical data from the analytical solution. It is the L^2 norm divided by n_x. Figure 6b shows the evolution in time of RMSE(t) for four values of P. For data within the time window $[0, 1200]$, values of $P < 0.015$ fit the data better at the early stage of the simulations. While values of $P > 0.015$ fit the data better at later stage. In this situation, the value of P should be chosen depending on whether high accuracy is desired to be achieved at the early or at the later stage of the simulations. The value of P could be also chosen in such way to accomplish the same order of error along the whole simulation. For example, by setting $P = 0.015$ during the time window $[0, 1200]$, the maximum error is the same at the early as well as at the later stage when approaching $t = 1200$. However, the value of $P = 0.012$ is used in other figures because it fits better the data at the early stage when $t < 800$.

Figure 6c reports the evolution in time of the concentration on either side of the membrane c_in and c_out. The solid lines are the analytical solution with $P = 0.012$. At the early stage, the theoretical solution is below the numerical data and around $t = 600$ the two curves cross each other, which explains the lower value of RMSE(t) observed in Fig. 6b. Beyond $t = 600$, the numerical solution tends faster to the equilibrium value, here $c_\mathrm{eq} = 0.5$, compared to the theory that decays very slowly.

Another way to appreciate the deviation of the numerical data from the theory is to compute the instantaneous local error,

$$\mathrm{Error}(x,t) = 100 \times \left| \frac{c_\mathrm{num}(x,t) - c_\mathrm{theo}(x,t)}{c_\mathrm{theo}(x,t)} \right| . \qquad (24)$$

This gives the local relative deviation of the computed solution from the theory. Along the overall simulation, every locally computed error is less than 10% before $t = 800$. But, as time evolves beyond $t = 800$, large deviations of the numerical solution from the analytical solution occur and lead to errors larger than 10%. These deviations may be explained by accumulated errors caused during the estimation of the concentration on either side of the membrane when using interpolation and extrapolation (Steps 2, 3 and 5 in Sec.4). The error goes back down when the system approaches and reaches equilibrium. For some applications, the most relevant details take place at the early stage of the simulations and not at long-term, as is the case for the drug delivery problems (see Sec. 6).

Figure 6e gives the approximate adequate value of P for every imposed value of R for three values of the diffusivity: $D = 0.167$ ($\tau_\mathrm{g} = 1$), 0.133 ($\tau_\mathrm{g} = 0.9$) and 0.1 ($\tau_\mathrm{g} = 0.8$). The larger is R, the lower is P. There exists an upper bound for the resistance R_max measured at $P = 0$ (impermeable membrane), beyond which unphysical solute transport takes place in the opposite direction to the concentration gradient, see Appendix A. Here, R_max depends on D: $R_\mathrm{max} = 0.5001$ ($D = 0.167$), $R_\mathrm{max} = 0.4445$ ($D = 0.133$) and $R_\mathrm{max} = 0.3750$ ($D = 0.1$). The minimum value of R can of course goes down to zero, and this corresponds to an infinitely permeable membrane ($P \to \infty$). Figure 6e shows that the value of R depends on P and also on the solute diffusivity D. For the same value of P, R should be reduced whenever D is decreased. The value of R also needs to be adapted to the grid resolution of the computational domain, as shown in Fig. 6f. It needs to be increased when refining the resolution. Figure 6f reports the convergence of the numerical data (the symbol points) to the theory (the solid line), obtained at the same scaled time $Dt/n_x^2 = 1/600$, for three grid resolutions.

All the reported results indicate that the prefactor R is apparently a function of the membrane permeability, the diffusion coefficient, in the inner and in the outer domains, and the grid resolution.

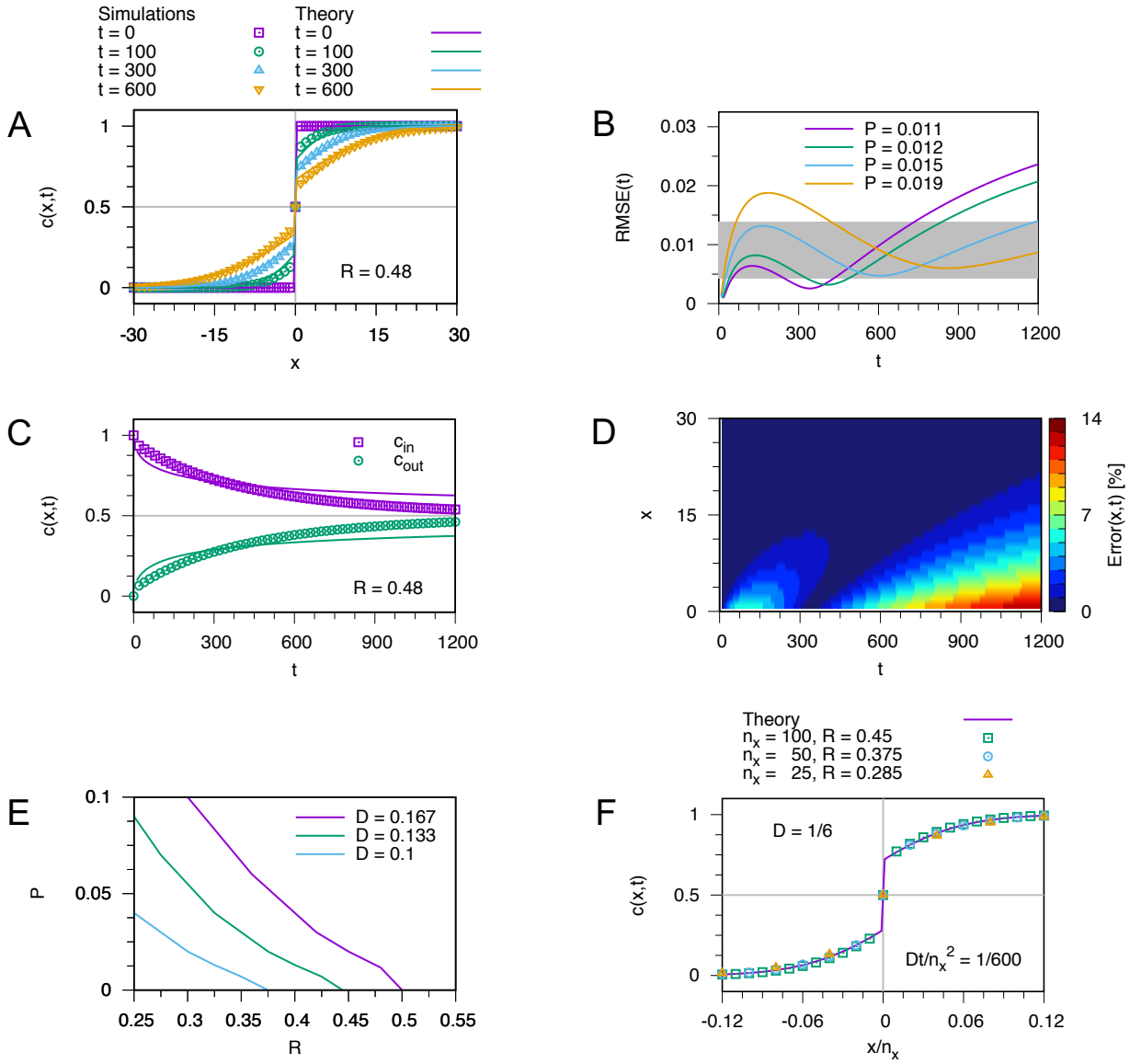

Fig. 6. (a) Fit of numerically computed concentration profiles $c(x,t)$ with their respective one-dimensional analytical solution at different elapsed times t, numerical data are represented with symbols and the analytical solution with solid lines, the fit parameter is set to $P = 0.012$, (b) Evolution in time of the fit quality measured by the RMSE Eq. 23, (c) Evolution in time of the concentration on either side of the membrane c_{in} and c_{out}, (d) Temporal evolution of the local relative error of the computer code Error(x,t) Eq. 24, (e) The adequate fitting parameter P versus the resistance R for three values of the diffusivity D, (f) Convergence of the numerical data to the analytical solution, at a given scaled time, using three different grid resolutions.

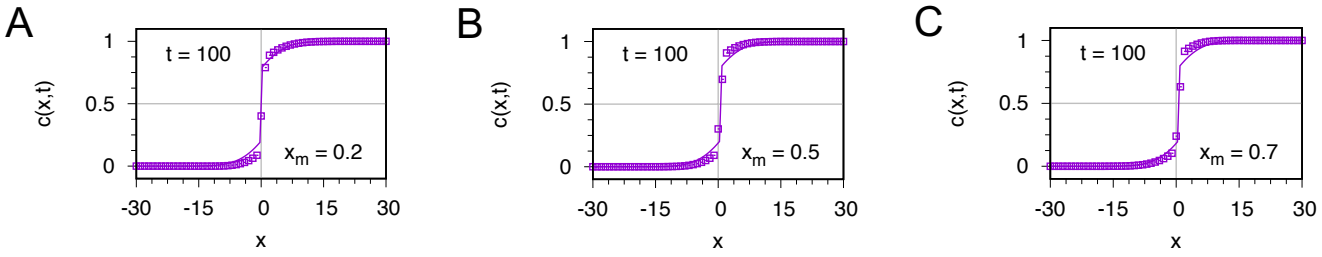

Fig. 7. Numerically computed concentration profiles $c(x,t)$ with symbols, and the expected analytical solution with solid lines, at time $t = 100$ and for three arbitrary off-lattice positions of the planar membrane: $x_{\text{m}} = 0.2$, $x_{\text{m}} = 0.5$ and (a) $x_{\text{m}} = 0.7$. The resistance is $R = 0.48$ that corresponds to the membrane permeability $P = 0.012$.

Another factor that is found to influence the quality of the computed solutions is the location of the membrane nodes. For the planar membrane that is parallel to the y-direction, if the x-coordinate of the nodes $x_{\rm m}$ is off-lattice then the fit is not anymore good, as shown in Fig. 7. However, the computer code still produces data that are reasonably comparable to the expected theoretical solution with a slight deviation. For the position of the membrane nodes in the y-direction, and for the configuration studied here, it is found not to influence the result. Exactly the same concentration profile is obtained whether the membrane nodes in the y-direction are on- or off-lattice.

Apparently the present numerical method introduces an excess numerical viscosity; and therefore, the obtained convincing results should be taken with precaution. Numerical viscosity is directly proportional to the gradients. At the early stages of simulation, gradients close to the membrane are high which means numerical viscosity is high. The numerical viscosity is qualitatively inversely proportional to diffusivity, which makes effective diffusivity small. For fixed R, permeability decreases with decreasing diffusivity (Fig. 6e). This is why for fixed R, the numerical results are matching well with analytical results for small permeability at early stages (high numerical viscosity) and large permeability at later stages (Fig. 6b). Similarly Fig. 6c shows, for a given R, numerical results are overestimating the analytical value as the numerical viscosity is high (small effective diffusivity), and at later stages it equilibrates faster as numerical viscosity goes to zero (high diffusivity). When the membrane is off the grid, as reported in Fig. 7, the numerical results are deviating from the analytical results. This can be attributed to the fact that the numerical viscosity is not symmetric on both sides of the membrane, and R used for these tests are from the calibration when the membrane is on the grid.

6 Applications

The proposed algorithm is used together with an LBM-based code to carry out two-dimensional simulations of solute release from particles. The particles are loaded initially with a solute that later on diffuses across the membrane towards the external surrounding domain $\Omega_{\rm out}$. The release rate is controlled by tuning the resistance R. The reported data are not meant to recover any existing experimental data, but rather to demonstrate the ability of the algorithm to reproduce the unsteady jump in the concentration at stationary and moving deformable membranes.

6.1 Solute release from a stationary rigid particle

A circular particle with radius 20, whose membrane is discretized with 125 points ($\Delta s = h = 1$), is placed at the center of a square domain of size 160×160. Zero-flux boundary conditions are set on the four edges of the square using the bounce-back boundary condition [17]. The initial concentration $c(x, y, t = 0)$ is set to $c_0 = 1$ inside the particle, and zero elsewhere. This situation mimics a reservoir loaded initially with a solute that later on diffuses across the membrane to the outer domain. The solvent is at rest and the particle does not move or undergo any shape deformation. Two situations are modeled: (i) a particle with finite membrane permeability $R = 0.48$, and (ii) a particle with infinite membrane permeability $R = 0$.

Figures 8a and 8b give the concentration $c(x, y, t)$ in the whole computational domain at time $t = 300$. For both cases, the diffusion is isotropic and symmetric with respect to the center of the particle. However, inside the particle in Fig. 8a is more reddish than in Fig. 8b. For the finite permeability case, the solute transport across the membrane is remarkably slowed down. This is why more solute is still retained inside the particle. The difference is more appreciable when reporting the solute concentration profiles along the radial coordinate r at different times, see Figs. 8c and 8d. The concentration shows smooth variation for the case of infinitely permeable particle, while it exhibits a steeper jump at the membrane for the restricted permeability case. The resulting discontinuity jump in the concentration at the membrane is unsteady and decays over time. The respective heights of the profiles are lower for the infinitely permeable membrane case. This means the solute leaks out quickly in absence of membrane resistance.

Figure 8e reports the release rate of the encapsulated solute, which is a global quantity and computed as [29],

$$\text{Release}(t) = 100 \times \left[\frac{M_{\rm in}(0) - M_{\rm in}(t)}{M_{\rm in}(0)} \right], \qquad (25)$$

where $M_{\rm in}(t)$ is the total mass of the solute inside the particle at time t,

$$M_{\rm in}(t) = \int_{\Omega_{\rm in}} c(x, y, t) \mathrm{d}x\mathrm{d}y, \qquad (26)$$

and whose initial value is $M_{\rm in}(0) = 1245$ and that is found to be conserved numerically

$$M_{\rm in}(0) = M_{\rm in}(t) + M_{\rm out}(t) \qquad (27)$$

with $M_{\rm out}(t)$ is the total mass of the solute outside the particle,

$$M_{\rm out}(t) = \int_{\Omega_{\rm out}} c(x, y, t) \mathrm{d}x\mathrm{d}y. \qquad (28)$$

By increasing R, the amount of the released solute is slowed down dramatically. However, all the curves converge into a single one when the concentration on both sides of the membrane tend to equilibrium,

$$c_{\rm in}(\mathbf{r}_{\rm m}, \infty) = c_{\rm out}(\mathbf{r}_{\rm m}, \infty) = c_{\rm eq}. \qquad (29)$$

This leads to a vanishing forcing term; and therefore, to the suppression of the resistance effect. The impact of the restricted permeability is noticeable at short time before the system reaches equilibrium.

This case study demonstrates the ability of the algorithm to reproduce the jump discontinuity in the solute concentration at a circular closed membrane, and its intuitively expected unsteady decay over time. To the best

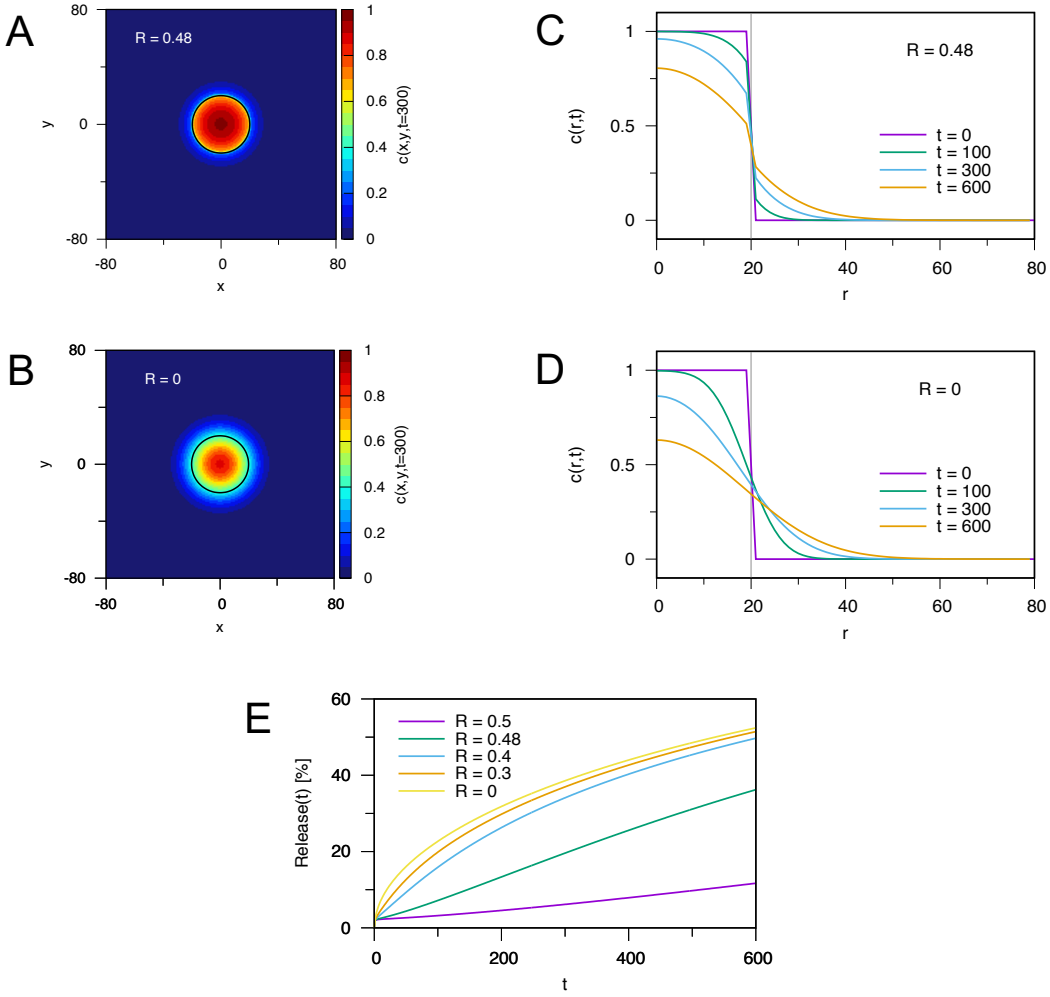

Fig. 8. Comparison between the solute concentration fields $c(x, y, t)$ at time $t = 300$ for two stationary rigid particles: one with a finite membrane permeability $R = 0.48$ (a) and the other with an infinite membrane permeability $R = 0$ (b). The contour with the black solid line represents the membrane location. (c,d) The corresponding numerically computed concentration profiles along the radial coordinate r, (f) Evolution in time of the solute release, Release(t), before the system reaches the equilibrium.

of the author's knowledge and as also stated by Sherk [30], no exact analytical solution exists for this case study that would allow to perform quantitative accuracy study. Nevertheless, the proposed algorithm reproduces the same qualitative features observed in the results computed by Huang *et al.* [11] and Sherk [30] using the IBM.

6.2 Solute release from a moving fluid-filled particle

The proposed algorithm can also handle problems that deal with fluid-solute-structure interaction. Here, the algorithm is used to handle the solute jump boundary condition at a moving deformable membrane of a fluid-filled particle. This is encountered, for example, in the following situations: oxygen uptake and delivery by red blood cells or drug delivery by liposomes and capsules. The interactions between different elements of such multiphysics problem are summarized in Fig. 9. The flow transports and deforms the membrane, which in its turn disturbs back the flow at its vicinity. The membrane is assumed to be inextensible and resists to bending. All the details about the fluid-structure two-way coupling via the IBM and the mechanics of the particle membrane have already been reported by the same author in Refs. [31–34], which have been confirmed by other research groups [35–37]. Here, the fluid flow and the mass transfer are both computed with the LBM [29,38].

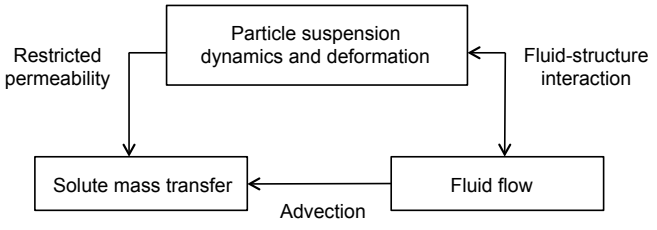

Fig. 9. Interactions between different elements of the fluid-solute-structure interaction problem studied in Sec. 6.2.

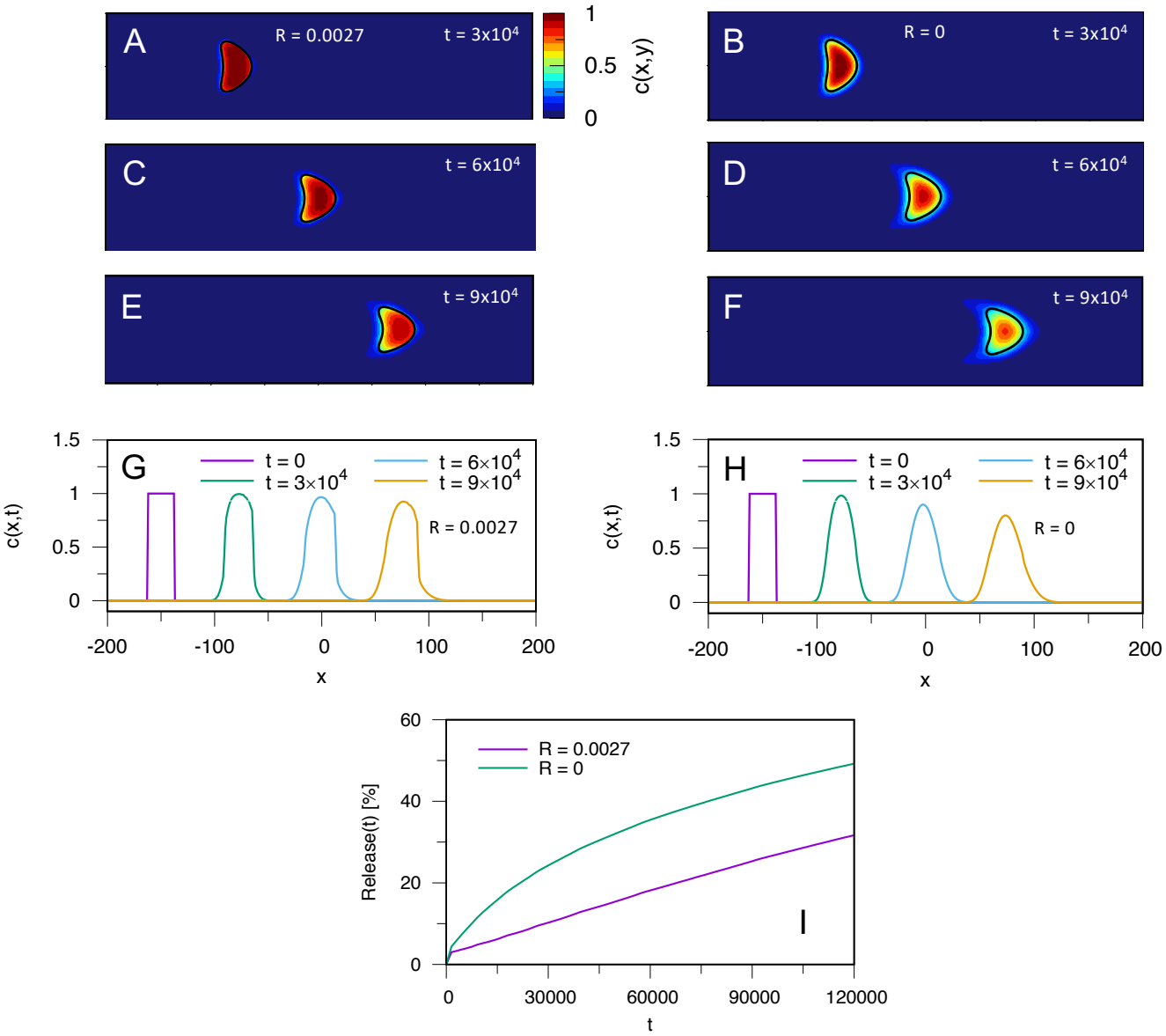

Fig. 10. Solute concentration inside and around a deformable fluid-filled particle moving in a channel at three different times: (a,c,e) a particle with a membrane resistance $R = 0.0027$, and (b,d,f) a particle with an infinitely permeable membrane $R = 0$, the contour with the black solid line represents the membrane location, the fluid flow is from left to right, (g,h) The corresponding solute concentration profiles in the x-direction, (k) The solute release Release(t) for the two cases.

The originality of the present work consists in implementing an explicit coupling between the membrane dynamics and the restrictive solute transport across it. The membrane is assumed to be permeable to solute, with finite permeability, and strictly impermeable to solvent. All these considerations are accomplished numerically via the proposed algorithm (Sec. 4). Two recent works have used only one-way coupling, *i.e.*, absence of any explicit mass transfer boundary condition on the membrane [29,39]. These studies are limited to situations where the membrane has an extremely large permeability [1], while the presently proposed algorithm handles the general case of membranes with finite permeability.

In the present application, two cases are performed to appreciate the impact of including a resistance to the membrane permeability. For these simulations, the computational domain is a rectangle of size 400×100. Periodic boundary conditions, for both the flow and the mass transfer, are set at the inlet and the outlet of the channel. The particle membrane is discretized with 120 points uniformly separated by $\Delta s = h = 1$, which is suitable for the IBM. The initial concentration $c(x, y, t = 0)$ is $c_0 = 1$ inside the particle, and zero elsewhere. The particle has initially an elliptical shape with long and minor axes 26.58 and 12.04, respectively. It is initially placed at the position $(-150, 0)$. The particle moves and deforms under the action of the flow, while conserving both its enclosed area

and its membrane perimeter due to its inextensibility. The Reynolds number is $\mathrm{Re} = UR_0/\nu = 0.1$, where U is the flow velocity at the channel centerline in absence of the particle, R_0 the effective size of the particle and ν the kinematic viscosity of the suspending fluid. The Schmidt number is $\mathrm{Sc} = \nu/D = 1000$. The capillary number is $\mathrm{Ca} = \eta U R_0^2/E_\mathrm{B} = 100$, where η is the dynamic viscosity and E_B the bending elastic modulus of the membrane.

The snapshots in Figs 10 show the concentration of the solute inside and outside the moving deformable particle for both cases: with (Figs 10a, 10c and 10e) and without membrane resistance (Figs 10b, 10d and 10f). The flow direction is from left to right. The particle moves while it undergoes shape deformation and it releases its encapsulated solute. All the snapshots have the same color bar range $[0, 1]$. Again, it is remarkable how finite permeability slows down the leakage of the solute from inside the particle to its surrounding flowing fluid. This is more visible when comparing side-by-side the concentration profiles (Figs. 10g and 10h) for both cases. The concentration profiles move due to advection. They exhibit a sharp jump in the concentration, in particular at the front of the particle, for the case of the membrane with resistance (Fig. 10g), and their heights are lower for the case without resistance (Fig. 10h). The profiles have almost a Gaussian like shape for the membrane with infinite permeability. The sum of the total mass of the solute, inside and outside the particle, is conserved for both cases during the simulations, and it equals the initial value $M_\mathrm{in}(0) = 1005$. The membrane resistance enhances the retention of the encapsulated solute. Fig. 10i gives the solute release rate for both cases: with and without resistance. A non-zero value of the resistance restricts the solute transport across the membrane, which slows down the overall release.

Again, this case study demonstrates the effectiveness of the proposed algorithm to reproduce the jump in the concentration at membranes of even moving deformable particles. There is no available comparable data as shown in Fig. 10 obtained by other theoretical, numerical or experimental methods for quantitative comparison purposes. The application situations computed, for example, in Refs. [7,12,14] are for membranes with different mechanics and different type of mass transfer boundary conditions. Here, the solute transport across the membrane takes place only by diffusion and the membrane is perfectly non-permeable to solvent.

7 Concluding remarks

The proposed algorithm is simple to implement in any existing code, which is based on the lattice Boltzmann method. It uses an *ad hoc* forcing term that captures the known behavior of solute transport across membranes with restricted permeability. It recovers the unsteady jump in the concentration at the membrane and its decay over time towards equilibrium. The computer code including the algorithm is validated with one-dimensional analytical solution for the planar membrane case given by Crank [25]. The instantaneous concentration profiles are perfectly computed for membranes with discretization nodes located exactly on-lattice, and with a slight deviation for off-lattice positions. Moreover, the algorithm can handle problems that involve fluid-solute-structure interaction and particles with arbitrary shapes. These features have been demonstrated by two case studies: mass transfer from a stationary rigid particle and from a moving deformable fluid-filled particle. The obtained results are qualitatively comparable to the ones reported in Refs. [11,30] for the case of a circular particle at rest, and as qualitatively expected for a moving deformable particle.

The concentration at the vicinity of either side of the membrane could be described by combining together, for example, Eq. 1 and Eq. 22 for the concentration on the inner side of the membrane. Under static conditions, the instantaneous adopted concentration on either side of the membrane is controlled by the membrane permeability, the Henry's partition coefficient (set to unity along this study), the diffusivities and the concentration derivatives on either side of the membrane (this later is sensitive to the degree of grid refinement), and by the equilibrium concentration that depends on other factors such as the size of the donor and the receptor domains and the area of the membrane. Under dynamical conditions, the concentration on either side of the membrane is expected to be controlled by the solvent speed as well. By taking into account all these considerations, $-R\left[c_\mathrm{in}(\mathbf{r}_\mathrm{m}, t) - c_\mathrm{out}(\mathbf{r}_\mathrm{m}, t)\right]$ is set phenomenologically as the leading-order term for the forcing term in this study. However, this approximation has revealed to not be accurate enough despite its success to reproduce the expected physics. The prefactor R lumps the effect of the diffusivity and the concentration derivatives on either side of the membrane in a non-trivial way, and this leads to non-constant effective membrane permeability that depends non-linearly on R. This suggests that the forcing term should be corrected by additional terms in order to enhance the algorithm accuracy.

The open actual issue with the proposed algorithm is the lack of physical meaning and a mathematical derivation of the *ad hoc* introduced numerical prefactor R. This can be seen as the membrane resistance, which is usually defined as the inverse of the permeability $\frac{1}{P}$ (or $\frac{\delta_\mathrm{m}}{P}$ for membranes with non-zero thickness δ_m). However, this is not the case here. R is a function of the permeability, the diffusivity and the grid resolution. Nevertheless, the computer code with the proposed algorithm and the *ad hoc* incorporation of the forcing term captures the essential physics of solute transport across stationary and moving deformable membranes with finite permeability. A computer code incorporating the proposed algorithm could be used as a numerical tool by pharmaceutists to predict qualitatively, at the moment, drug release kinetics from particles with membranes presenting solute finite permeability under blood flow conditions. Combination with other advanced LBM schemes may enhance further the stability and the accuracy of the algorithm. Extension to three-dimensional space and adaptation to heat transfer problems are expected to be straightforward.

Acknowledgments

The author thanks the anonymous referees for their useful comments and suggestions.

Appendix A.

Figure. 11 shows the evolution in time of the computed concentration profile for the case of a planar membrane when the numerical value of the resistance R is set beyond the upper bound mentioned in Sec. 5. All the numerical set-up and parameters are the same as those used in Sec. 5. The threshold is $R_{\max} = 0.5001$ for the solute diffusion coefficient $D = 0.167$, see Fig. 4e. The obtained concentration profiles when setting $R = 0.51$ show an unphysical scenario that corresponds to solute diffusion from the outer towards the inner domain, *i.e.*, in the opposite direction to the concentration gradient. When R is set to its maximum, $R_{\max} = 0.5001$, the concentration profile evolves to a steady profile shown in Fig. 12 and that reflects the case of a solute impermeable membrane.

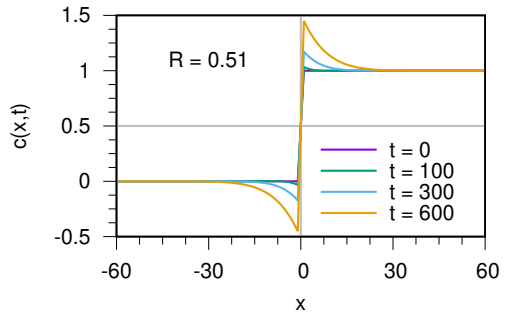

Fig. 11. The resulting computed concentration profiles normal to a planar membrane when setting the resistance to $R = 0.51$, which is beyond the upper bound $R_{\max} = 0.5001$ for the diffusion coefficient $D = 0.167$.

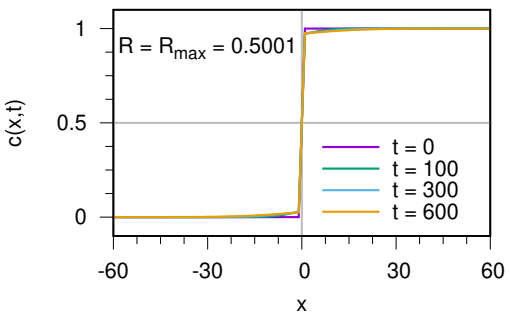

Fig. 12. The computed concentration profiles normal to a planar membrane when setting the resistance R to its maximum $R_{\max} = 0.5001$ with the diffusion coefficient $D = 0.167$.

References

1. R. L. Fournier, *Basic transport phenomena in biomedical Engineering*, Taylor & Francis, Philadelphia, PA, USA, 1999.
2. G. A. Truskey, F. Yuan, and D. F. Katz, *Transport phenomena in biological systems*, Pearson, Upper Saddle River, NJ, USA, 2010.
3. W-C. Wang, *A jump condition capturing finite difference scheme for elliptic interface problems*, SIAM Journal on Scientific Computing 25 (2004), pp. 1479-1496.
4. R. I. Hickson, S. I. Barry, G. N. Mercer, and H. S. Sidhu, *Finite difference schemes for multilayer diffusion*, Mathematical and Computer Modelling 54 (2011), pp. 210-220.
5. A. T. Layton, *Modeling water transport across elastic boundaries using an explicit jump method*, SIAM Journal on Scientific Computing 28 (2006), pp. 2189-2207.
6. P. G. Jayathilake, Z. Tan, B. C. Khoo, and N. E. Wijeysunder, *Deformation and osmotic swelling of an elastic membrane capsule in Stokes flows by the immersed interface method*, Chemical Engineering Science 65 (2010), pp. 1237-1252.
7. P. G. Jayathilake, G. Liu, Z. Tan, and B. C. Khoo, *Numerical study of a permeable capsule under Stokes flows by the immersed interface method*, Chemical Engineering Science 66 (2011), pp. 2080-2090.
8. Z. Li and K. Ito, *The immersed interface method: Numerical solutions of PDEs involving interfaces and irregular domains*, SIAM, Philadelphia, PA, USA, 2006.
9. S. Miyauchi, S. Takeuchi, and T. Kajishima, *A numerical method for mass transfer by a thin moving membrane with selective permeabilities*, Journal of Computational Physics 284 (2015), pp. 490-504.
10. S. Miyauchi, S. Takeuchi, and T. Kajishima, *A numerical method for interaction problems between fluid and membranes with arbitrary permeability for fluid*, Journal of Computational Physics 345 (2017), pp. 33-57.
11. H. Huang, K. Sugiyama, and S. Takagi, *An immersed boundary method for restricted diffusion with permeable interfaces*, Journal of Computational Physics 228 (2009), pp. 5317-5322.
12. X. Gong, Z. Gong, and H. Huang, *An immersed boundary method for mass transfer across permeable moving interfaces*, Journal of Computational Physics 278 (2014), pp. 148-168.
13. C. S. Peskin, *The immersed boundary method*, Acta Numerica 479 (2002), pp. 479-517.
14. L. Yao and Y. Mori, *A numerical method for osmotic water flow and solute diffusion with deformable membrane boundaries in two spatial dimension*, Journal of Computational Physics 350 (2017), pp. 728-746.
15. D. A. Wolf-Gladrow, *Lattice-gas cellular automata and lattice Boltzmann models: An introduction*, Springer, Germany, 2000.
16. S. Succi, *The lattice Boltzmann equation for fluid dynamics and beyond*, Oxford University Press, Oxford, UK, 2001.
17. M. C. Sukop and D. T. Thorne, *Lattice Boltzmann modeling*, Springer, Berlin, Germany, 2006.
18. A. A. Mohamad, *Lattice Boltzmann method: Fundamentals and engineering applications with computer codes*, Springer, New York, 2011.
19. T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, *The Lattice Boltzmann Method: Principles and Practice*, Springer International Publishing, Switzerland, 2017.

20. C. K. Aidun and J. R. Clausen, *Lattice-Boltzmann method for complex flows*, Annual Review of Fluid Mechanics 42 (2010), pp. 439-472.
21. T. Zhang, B. Shi, Z. Guo, Z. Chai, and J. Lu, *General bounce-back scheme for concentration boundary condition in the lattice-Boltzmann method*, Physical Review E 85 (2012), 016701.
22. T. Inamuro, M. Yoshino, H. Inoue, R. Mizuno, and F. Ogino, *A lattice Boltzmann method for a binary miscible fluid mixture and its application to a heat-transfer problem*, Journal of Computational Physics 179 (2002), pp. 20-215.
23. K. Guo, L. Li, G. Xiao, N. Auyeung, and R. Mei, *Lattice Boltzmann method for conjugate heat and mass transfer with interfacial jump conditions*, International Journal of Heat and Mass Transfer 88 (2015), pp. 306-322.
24. A. Kasparek, P. Lapka, and P. Furmanski, *Application of a coupled Lattice Boltzmann and Immersed Boundary Method for solving conjugated heat transfer problems*, Journal of Physics: Conference Series 745 (2016), 032024.
25. J. Crank, *The mathematics of diffusion*, Oxford University Press, 1975.
26. P. L. Bhatnagar, E. P. Gross, and M. Krook, *A Model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems*, Physical Review 94 (1954), 511
27. Y. H. Qian, D. d'Humières, and P. Lallemand, *Lattice BGK models for Navier-Stokes equation*, Europhysics Letters 7 (1992), 479
28. M. M. Dupin, I. Halliday, C.M. Care, L. Alboul, and L. L. Munn, *Modeling the flow of dense suspensions of deformable particles in three dimensions*, Physical Review E 75 (2007), pp. 066707
29. B. Kaoui, *Computer simulations of drug release from a liposome into the bloodstream*, European Physical Journal E - Soft Matter and Biological Physics 41, (2018), pp. 20
30. T. R. H. Sherk, *Numerical methods for simulating diffusion in cellular media*, Master of Science Thesis, University of Ontario Institute of Technology, 2011.
31. B. Kaoui, J. Harting and C. Misbah, *Two-dimensional vesicle dynamics under shear flow: effect of confinement*, Physical Review E 83 (2011), 066319.
32. B. Kaoui, T. Krüger, and J. Harting, *How does confinement affect the dynamics of viscous vesicles and red blood cells?* Soft Matter 8 (2012) 9246.
33. B. Kaoui, R. Jonk, and J. Harting, *Interplay between microdynamics and macrorheology in vesicle suspensions*, Soft Matter 10 (2014), 4735.
34. B. Kaoui and J. Harting, *Two-dimensional lattice Boltzmann simulations of vesicles with viscosity contrast*, Rheologica Acta 55 (2016), pp. 465
35. L. Shi, T-W. Pan, and R. Glowinski, *Deformation of a single red blood cell in bounded Poiseuille flows*, Physical Review E 85 (2012), 016307
36. I. Halliday, S. V. Lishchuk, T. J. Spencer, G. Pontrelli, and C. M. Care, *Multiple-component lattice Boltzmann equation for fluid-filled vesicles in flow*, Physical Review E 87 (2013), 023307
37. T. M. Fischer and R. Korzeniewski, *Threshold shear stress for the transition between tumbling and tank-treading of red blood cells in shear flow: dependence on the viscosity of the suspending medium*, Journal of Fluid Mechanics 736 (2013), 351-365
38. B. Kaoui, *Flow and mass transfer around a core-shell reservoir*, Physical Review E 95 (2017), 063310
39. G. Kabacaoğlu, B. Quaife, and G. Biros, *Quantification of mixing in vesicle suspensions using numerical simulations in two dimensions*, Physics of Fluids 29 (2017), 021901