UPPSALA
UNIVERSITET

Stochastic models in molecular biology

Siyang Wang

Project in Computational Science: Report

January 2012

PROJECT REPORT

Department of Information Technology

Abstract

The diffusion processes of molecules in a living cell are simulated and analyzed based on a multiscale model, which is a combination of a microscopic model and a mesoscopic model. In a microscopic model the exact position of each individual molecule is followed. A coarser level of modeling is the mesoscopic model where the state of the system is given by the copy number of molecules of each species in voxels in a discretized space. The relationship between these two models and the methods to couple them will be studied in simulations.

Key words: stochastic simulation, diffusion processes, multiscale models

1 Introduction

There are two basic chemical processes of the molecules in a living cell: diffusion and reaction. In this report, the diffusion processes are analyzed and studied. Diffusion can be considered as random migration of molecules due to the thermal energy [3]. Generally, there are two fundamental approaches to the mathematical modeling of chemical diffusions: deterministic models which are based on partial differential equations (PDEs) for the concentrations of the molecules involved and the stochastic simulations with a multiscale model.

For the first approach, it is assumed that there is a large number of molecules in the system. However, it does not always hold in a living cell as some species appear in a very small copy number, thus an error would be the result.

Compared with the first approach, a stochastic model gives a more accurate result. In general, there are two distinct levels: microscopic level and mesoscopic level. At a microscopic level the diffusion process is based on Brownian motion (molecular-based) and it is simulated by the calculation of trajectories of the molecules. A coarser level of modeling is the mesoscopic level where the state of the system is given by the number of molecules of each species in cubes in a discretized space (cube-based).

The simulation at a microscopic level is accurate because each molecule is tracked individually by an exact position and free to migrate in a continuous domain. The position, which is denoted by the coordinate in 3D, is updated in a small time step. However, it is also expensive and sometimes it is a waste of time if the precise positions of the molecules are not so important. Therefore, it is only preferred when the concentrations are very small.

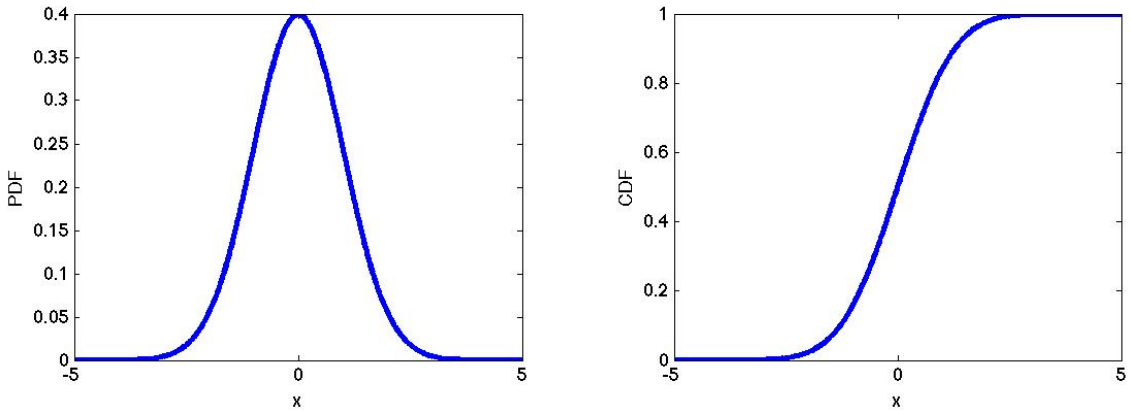
The simulation at a mesoscopic level is characterized by a discretization of the spatial domain into cubes. The state of the system is considered to be the number of molecules in the cubes. A molecule is free to migrate in the form of a discrete jump from one cube to another. The molecules in the same cube are indistinguishable. Thus, the cube-based model does not represent the real trajectories of the molecules. It is easier to implement and preferred when the concentrations are large.

In this report, we propose a model which couples the simulation at both microscopic level and mesoscopic level. The whole domain Ω is divided into two parts, Ω_M and Ω_C, with an interface I between them. In order to find the best way to couple these two models we need to study and summarize the molecular-based and cube-based simulation algorithms. We are also interested in the way that a molecule migrates across the boundary I.

2 Molecular-based modeling

The molecular-based model is based on Brownian dynamics. The approach to simulate such a system is to use a time-driven algorithm by choosing a small time step, say Δt, and updating the position of each molecule by the discretized version of stochastic differential equations [5]. Here we need to generate random numbers which are normally distributed with mean 0 and variance 1. The probability density function (PDF) is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-u)^2}{2\sigma^2}} \tag{1}$$



(a) (b)

Figure 1: (a) The PDF of the normal distribution with mean 0 and variance 1 (b) The CDF of the normal distribution with mean 0 and variance 1

and the cumulative distribution function (CDF) is

$$F(x) = \frac{1}{2}[1 + \text{erf}(\frac{x-u}{\sqrt{2}\sigma})] \tag{2}$$

shown in Figure 1.

We use these random numbers to calculate the new positions of the molecules at the next time step. Once the time step Δt is selected, the new positions are also affected by a diffusion constant D. The diffusion constant D is proportional to the expected squared velocity of the diffusing molecules, which depends on the temperature, viscosity of the fluid in the living cell and the size of the molecules according to the Stokes-Einstein relation [11].

Another important issue to be considered is that the diffusion occurs in a bounded domain, a cube with side length L. It leads to that the domain in the simulation must have a boundary and suitable boundary conditions which could be reflective, absorbing or reactive [4]. In this report, we use reflective boundary conditions.

We simulate up to a certain time and then sum up the number of molecules along y-direction and z-direction, study the distribution along x-direction which should converge to the mean value with the assumption that we have a large amount of molecules. The mean value is the solution of the partial differential equation for the density φ

$$\frac{\partial \varphi}{\partial t} = D\frac{\partial^2 \varphi}{\partial x^2}, \quad 0 < x < L \tag{3}$$

with the initial condition $\varphi(x, 0) = \delta(k - x)$ and boundary condition $\varphi_x(0, t) = \varphi_x(1, t) = 0$ where δ is the Dirac distribution at the origin and k is the x-coordinate of the molecule at time 0. The solution can be found by applying the method of separation of variables to (3) as follows:

$$\varphi = \frac{1}{L} + \sum_{n=1}^{\infty} \frac{2}{L} \cos(kn\pi) \cos\frac{n\pi x}{L} e^{-D(\frac{n\pi}{L})^2 t}. \tag{4}$$

We use (4) to test the convergence of our algorithms. The procedure of the molecular-based modeling is summarized in Algorithm 1.

3 Cube-based modeling

in this section, Gillespie's Stochastic Simulation Algorithm [8] is described. The main idea of the cube-based modeling is to use an event-driven algorithm in a discretized space as simulation

Algorithm 1 Molecular-based algorithm

(a) Generate three normally distributed (with zero mean and unit variance) random numbers ξ_x, ξ_y and ξ_z.

(b) Compute the position of the molecle at time $t + \Delta t$ by

$$X(t + \Delta t) = X(t) + \sqrt{2D\Delta t}\xi_x \tag{5}$$
$$Y(t + \Delta t) = Y(t) + \sqrt{2D\Delta t}\xi_y \tag{6}$$
$$Z(t + \Delta t) = Z(t) + \sqrt{2D\Delta t}\xi_z \tag{7}$$

(c) If a molecule moves across the boundary in the x-direction, then reflect it back at the same time step as follows

If $X(t + \Delta t)$ computed by (1.1) is less than 0, then
$X(t + \Delta t) := -X(t + \Delta t)$
If $X(t + \Delta t)$ computed by (1.1) is greater than L, then
$X(t + \Delta t) := 2L - X(t + \Delta t)$

Reflect the molecule back when it moves across the boundary in the y-direction or z-direction using the method above.

(d) Continue with step **(a)** for time $t + \Delta t$ until the desired end of simulation.

---

domain. Assume that the domain of interest Ω is a cube with side L and all the molecules are expected to move inside the domain. Then the domain is discretized into many small cubes with equal side length, say h, which should be chosen appropriately to make sure that $n = \frac{L}{h}$ is an integer. Thus, we have n^3 small cubes. Molecules do not have exact positions as they do in molecular-based models. Instead, they are only assigned to be in one of the cubes at a given time and allowed to jump from one cube to another based on availability. We denote the number of molecules in the i-th cube by $A(x, y, z)$ where $x, y, z = 1, 2, ..., n$ and $i = x + (y - 1)n + (z - 1)n^2$.

3.1 Available cubes for jump

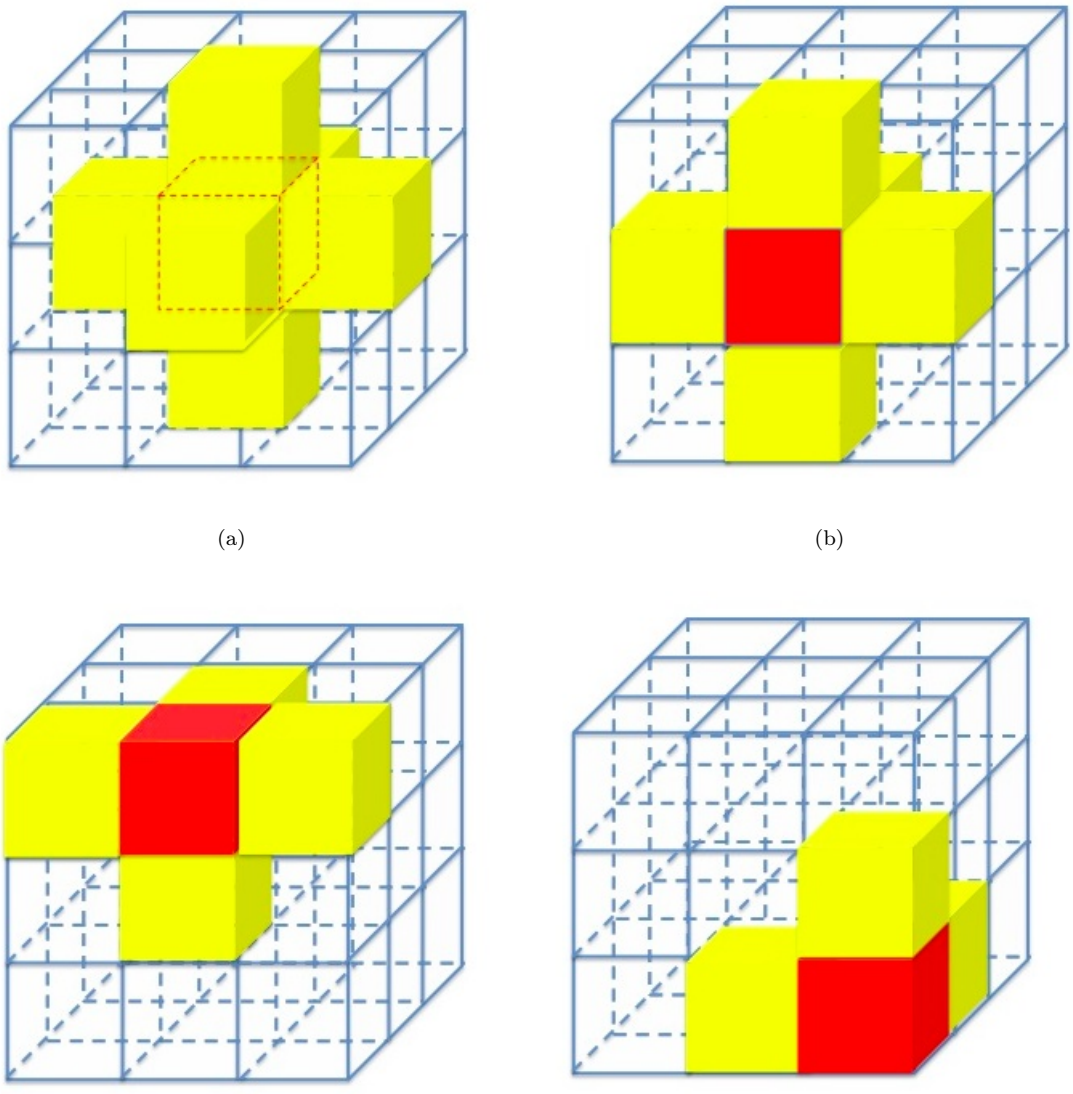
Now we determine the number of available cubes to which a molecule can jump according to its current position.

(i) If a molecule is currently in a cube which lies on none of the six faces of Ω, then there are six available cubes and it can jump to the cube with the direction of up, down, left, right, front or back. See Figure 2(a).

(ii) If a molecule is currently in a cube which lies on only one of the six faces of Ω, then there are five available cubes. See Figure 2(b).

(iii) If a molecule is currently in a cube which lies on two of the six faces of Ω, then there are four available cubes. See Figure 2(c).

(iv) If a molecule is currently in a cube which lies on three of the six faces of Ω, then there are three available cubes. See Figure 2(d).



(a) (b)

(c) (d)

Figure 2: Four cases of the number of available cubes: A molecule is currently in the red cube with whose available cubes are illustrated by the yellow cubes

3.2 Propensity functions

In the event-driven algorithm, only one molecule is allowed to move when an event occurs. The direction is selected with the help of propensity functions. We denote them by $\alpha_i(t)$, where $i = 1, 2, ..., n^3$. At time t, $\alpha_i(t)$ is the probability that the diffusion event occurs in the $i-th$ cube in the time interval $[t, t+dt)$. Here we assume that all the jumps have the same rate constant d. It is calculated by $d = \frac{D}{h^2}$ where D is the diffusion constant and h is the side length of a small cube. The propensity functions at time t are computed by $\alpha_i(t) = A_t(x, y, z)d$ where $x, y, z = 1, 2, ..., n$ and $i = x + (y-1)n + (z-1)n^2$. Then compute:

$$\alpha_1 = d\sum_{z=1}^{n-1}\sum_{y=1}^{n}\sum_{x=1}^{n} A(x, y, z) \tag{8}$$

$$\alpha_2 = d\sum_{z=2}^{n}\sum_{y=1}^{n}\sum_{x=1}^{n} A(x, y, z) \tag{9}$$

$$\alpha_3 = d\sum_{z=1}^{n}\sum_{y=1}^{n}\sum_{x=2}^{n} A(x, y, z) \tag{10}$$

$$\alpha_4 = d\sum_{z=1}^{n}\sum_{y=1}^{n}\sum_{x=1}^{n-1} A(x, y, z) \tag{11}$$

$$\alpha_5 = d\sum_{z=1}^{n}\sum_{y=2}^{n}\sum_{x=1}^{n} A(x, y, z) \tag{12}$$

$$\alpha_6 = d\sum_{z=1}^{n}\sum_{y=1}^{n-1}\sum_{x=1}^{n} A(x, y, z) \tag{13}$$

$$\alpha_7 = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 \tag{14}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ and α_6 correspond to diffusion towards the direction of up, down, left, right, front and back, respectively. α_7 is the propensity function of diffusion in the whole system.

3.3 The time at which the next difussion event occurs

The time between two diffusion events is calculated by the formula

$$\Delta\tau_C = \frac{1}{\alpha_7}\ln(\frac{1}{r_1}) \tag{15}$$

where α_7 is calculated by (14) and r_1 is a uniformly distributed number in (0,1). This formula is verified in [5].

3.4 Selection of molecules

The decision about which molecule jumps and which direction it takes is made with the help of a uniformly distributed random number r in (0,1). Then we update the number of molecules in two cubes: one is the cube the molecule jumps from; the other is the cube the molecule jumps to. We summarize it as follows:

(i) If $r < \alpha_1/\alpha_7$, then find the first $A(i, j, k)$ such that

$$r < \frac{d}{\alpha_7}\sum_{z=1}^{k}\sum_{y=1}^{j}\sum_{x=1}^{i} A(x, y, z)$$

Then update the number of molecules at time $t + \Delta\tau_C$ by

$$A_{t+\tau}(i, j, k) = A_t(i, j, k) - 1$$
$$A_{t+\tau}(i, j, k+1) = A_t(i, j, k+1) + 1$$

(ii) If $r \geq \alpha_1/\alpha_7$ and $r < (\alpha_1 + \alpha_2)/\alpha_7$, then find the first $A(i,j,k)$ such that

$$r < \frac{1}{\alpha_7}(\alpha_1 + d\sum_{z=1}^{k}\sum_{y=1}^{j}\sum_{x=1}^{i} A(x,y,z))$$

Then update the number of molecules at time $t + \Delta\tau_C$ by

$$A_{t+\tau}(i,j,k) = A_t(i,j,k) - 1$$
$$A_{t+\tau}(i,j,k+1) = A_t(i,j,k+1) + 1$$

(iii) If $r \geq (\alpha_1 + \alpha_2)/\alpha_7$ and $r < (\alpha_1 + \alpha_2 + \alpha_3)/\alpha_7$, then find the first $A(i,j,k)$ such that

$$r < \frac{1}{\alpha_7}(\alpha_1 + \alpha_2 + d\sum_{z=1}^{k}\sum_{y=1}^{j}\sum_{x=1}^{i} A(x,y,z))$$

Then update the number of molecules at time $t + \Delta\tau_C$ by

$$A_{t+\tau}(i,j,k) = A_t(i,j,k) - 1$$
$$A_{t+\tau}(i,j,k+1) = A_t(i,j,k+1) + 1$$

(iv) If $r \geq (\alpha_1 + \alpha_2 + \alpha_3)/\alpha_7$ and $r < (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4)/\alpha_7$, then find the first $A(i,j,k)$ such that

$$r < \frac{1}{\alpha_7}(\alpha_1 + \alpha_2 + \alpha_3 + d\sum_{z=1}^{k}\sum_{y=1}^{j}\sum_{x=1}^{i} A(x,y,z))$$

Then update the number of molecules at time $t + \Delta\tau_C$ by

$$A_{t+\tau}(i,j,k) = A_t(i,j,k) - 1$$
$$A_{t+\tau}(i,j,k+1) = A_t(i,j,k+1) + 1$$

(v) If $r \geq (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4)/\alpha_7$ and $r < (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5)/\alpha_7$, then find the first $A(i,j,k)$ such that

$$r < \frac{1}{\alpha_7}(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + d\sum_{z=1}^{k}\sum_{y=1}^{j}\sum_{x=1}^{i} A(x,y,z))$$

Then update the number of molecules at time $t + \Delta\tau_C$ by

$$A_{t+\tau}(i,j,k) = A_t(i,j,k) - 1$$
$$A_{t+\tau}(i,j,k+1) = A_t(i,j,k+1) + 1$$

(vi) If $r \geq (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5)/\alpha_7$, then find the first $A(i,j,k)$ such that

$$r < \frac{1}{\alpha_7}(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + d\sum_{z=1}^{k}\sum_{y=1}^{j}\sum_{x=1}^{i} A(x,y,z))$$

Then update the number of molecules at time $t + \Delta\tau_C$ by

$$A_{t+\tau}(i,j,k) = A_t(i,j,k) - 1$$
$$A_{t+\tau}(i,j,k+1) = A_t(i,j,k+1) + 1$$

Jumps to the directions of up, down, left, right, front and back are implemented in (i), (ii), (iii), (iv), (v) and (vi), respectively. The procedure of the cube-based stochastic simulation is summarized in Algorithm 2. This is a straightforward implementation of Gillespie's Stochastic Simulation Algorithm (SSA). Faster alternatives can be found in [7] and [9].

Algorithm 2 Cube-based algorithm

(a) Generate two random numbers r_1 and r_2 which are uniformly distributed in (0, 1).

(b) Compute the propensity functions using (14).

(c) Compute the time $t + \Delta\tau_C$ at which the next diffusion occurs where $\Delta\tau_C$ is given by (15).

(d) Update the whole system with the help of the algorithm in (3.4).

(f) Continue with step **(a)** with $t + \Delta\tau_C$ until the desired end of simulation.

4 Algorithm from University of Oxford

In this section, a spatially hybrid model with the two-regime method (TRM) [6], is described and implemented in 3D. In this model, the domain of interest Ω is divided into two subdomains. One is the molecular-based domain, denoted by Ω_M and the other is cube-based domain, denoted by Ω_C. Molecules in both Ω_C and Ω_M are simulated according to the rules defined by their particular algorithms. The illustration of this model is shown in Figure 3(a). There is also an interface between Ω_M and Ω_C, denoted by I. The molecules in Ω_M are allowed to move to Ω_C via I and vice versa. As we already have the algorithms for molecules moving inside each subdomain, we will shortly show how to treat the molecules if they move across the interface.

The simulations of Ω_M and Ω_C are performed continuously with the help of random numbers. The process is illustrated in Figure 3(b). The procedure is summarized in Algorithm 3.

The algorithm proceeds by repeating steps **(c)** and **(d)**, which computes C-events and M-events, respectively. At each M-event, the new molecules that migrated from Ω_C to Ω_M are introduced by placing them at a distance x from the interface I that is sampled from the probability distribution $f(x)$ given by

$$f(x) = \sqrt{(\frac{\pi}{4D\Delta t})}\text{erfc}(\frac{x}{\sqrt{4D\Delta t}}) \tag{17}$$

The probability density function (PDF) and the cumulative distribution function (CDF) are shown in Figure 4. It is computationally expensive to sample from $f(x)$. Instead, x can be computed approximately as [1]:

$$x = \sqrt{2D\Delta t}\frac{0.729614P - 0.70252P^2}{1 - 1.47494P + 0.484371P^2} \tag{18}$$

where P is a uniform deviate between 0 and 1.

The y and z coordinates are sampled from the uniform distribution.

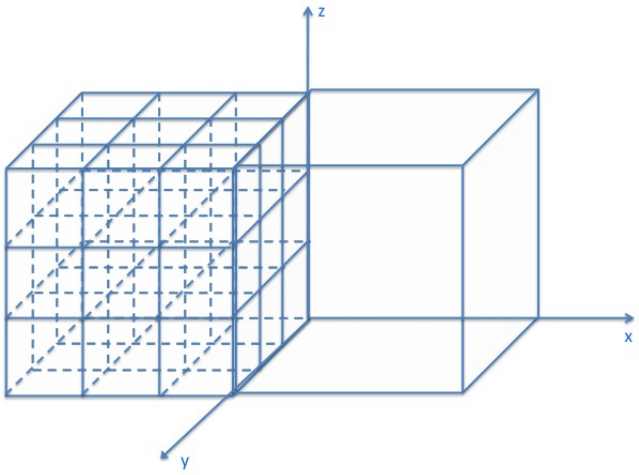
The propensity for molecules to migrate from cubes adjacent to the interface I into Ω_M is given by $\Phi D/h^2$ [6] per molecule. The coefficient Φ is the change in the propensity of migration to make the molecular flux over the interface I consistent with diffusion. Φ is determined by

$$\Phi = \frac{2h}{\sqrt{\pi D\Delta t}} \tag{19}$$

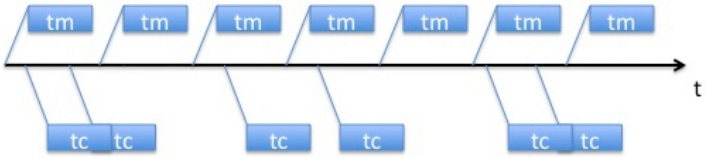
where Δt is the fixed time increment defined for updating the molecules in Ω_M.

For the selection of molecules moving from Ω_M to Ω_C via the interface, an error proportional to the net flux over the interface occurs if one allows transfer of molecules based on whether or not they appear on the other side of the interface [6]. Instead, we use P_m to be the probability of the migration from Ω_M to Ω_C. This probability has been determined in [2] as

$$P_m = \exp(\frac{-\Delta x_{old}\Delta x_{new}}{D\Delta t}). \tag{20}$$

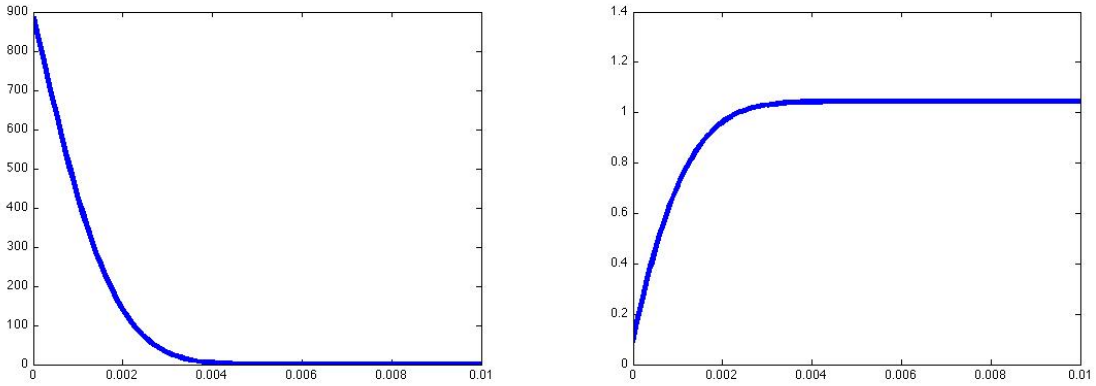


(a)



(b)

Figure 3: (a) Illustration of the domain in the multiscale model (b) Illustration of time update



(a) (b)

Figure 4: (a) The PDF of f(x) (b)The CDF of f(x)

Algorithm 3 The two-regime method (TRM) from University of Oxford

(a) Initialize the number of molecules in the cubes in Ω_C and the positions of molecules that are in Ω_M at $t = 0$.

(b) Choose Δt, the time between two consecutive updates of the molecular-based regime (M-events) in Ω_M. Use

$$\tau_{i,j} = \frac{1}{\alpha_{i,j}} \ln(\frac{1}{r_{i,j}}) \tag{16}$$

to calculate $\tau_{i,j}$, the times at which the next migratory events (C-events) will take place in Ω_C (or are initiated in Ω_C for jumps over the interface I). Set $t_M = \Delta t$ and $t_C = \min(\tau_{i,j})$. The minimum is taken over all $i = 1, 2...M$ and $j = 1, 2, ...K$ where M is the number of available cubes around the i-th cube and K is the number of cubes in Ω_C.

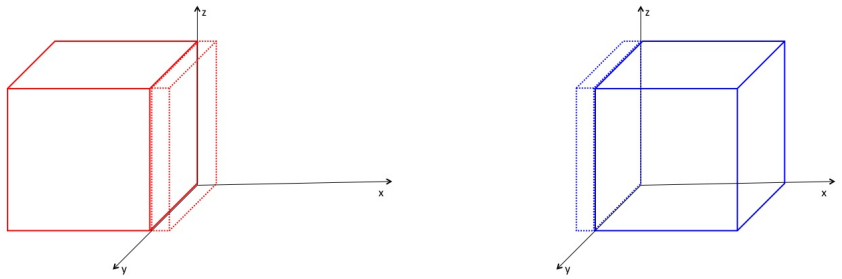
(c) if $t_C \leq t_M$, then the next C-event occurs:

- Update the current time $t = t_C$.
- Change the number of molecules in Ω_C to reflect the specific C-event that has occured. If this event is such that a molecule leaves Ω_C bound for Ω_M, then compute its initial position in Ω_M according to the method described in the text and remove it from the corresponding cube adjacent to the interface I. Calculate the time at which the next migratory event occurs for the current C-event by (16).
- Set $t_C := t_C + \min(\tau_{i,j})$.

(d) if $t_M < t_C$, then the next M-event occurs:

- Update the current time $t = t_M$.
- Update the positions of all molecules in Ω_M according to equations (5)$\sim$(7).
- Initialize all molecules which migrated from Ω_C to Ω_M during previous C-events at positions computed in step **(c)**, using (17).
- Absorb all molecules that interact with the interface I from Ω_M (excluding those just initiated) into the cube which is closest to their last calculated positions. The molecules moving across the interface are selected with the help of (20).
- Calculate the time at which the next migratory event occurs for the current C-event by (16).
- Update $t_M := t_M + \Delta t$ and, if there are any molecules moving from Ω_M into Ω_C, set $t_C := t_C + \min(\tau_{i,j})$.

(f) Repeat steps **(c)** and **(d)** until the desired end of the simulation



(a) (b)

Figure 5: Illustration of splitting domain Ω into two subdomains Ω_C and Ω_M

5 Multiscale modeling

5.1 Initial approach

Algorithm 3 performs rather satisfactory. However, it is not efficient enough as it takes a long time for simulation in its present implementation. In this section, we propose a more efficient algorithm which is still accurate and converges well, the splitting domain algorithm [10].

The domain of interest is the same as shown in Figure 3. We start the simulation at time $t = 0$ and choose a time step $\Delta\tau$. In each time step, we simulate the subdomains separately. Firstly, freeze Ω_M, which means that the molecules in Ω_M are not allowed to move, and update Ω_C as described in Algorithm 2. In order to allow molecules to move across the interface I, we add an empty layer of cubes adjacent to the interface, which contains n^2 cubes with the same side as the ones in Ω_C, see Figure 5(a). There are no molecules in these extra cubes at the beginning of each time step, but the molecules in other cubes can jump there by diffusion. The molecules in the extra cubes at the end of the current time step will be initialized and given the exact positions in Ω_M in the next time step. In Figure 3(a) one can see that the interface between Ω_C and Ω_M is $x = 0$. Thus, the y-coordinates and z-coordinates can be sampled from a uniform distribution and rescaled linearly to the corresponding intervals. There are many ways to sample the x-coordinates, such as from a uniform distribution and from $f(x)$ in (17). We implement both of them and compare the results later in this report.

Secondly, freeze Ω_C, which means that all the molecules in Ω_C are not allowed to move, and update Ω_M as described in Algorithm 1. In order to allow molecules to move across the interface I, we add an empty layer of cubes adjacent to the interface, see Figure 5(b). Here we must be careful to choose the time Δt in equations (5)$\sim$(7) after which the positions of all molecules in Ω_M are updated. In order to keep a balance between accuracy and efficiency, we set $\Delta t = \frac{1}{10}\Delta\tau$ which means that the positions of all molecules in Ω_M are updated 10 times in each time step $\Delta\tau$. We also need to make sure that the molecules moving across the interface can not go too far away from the interface into Ω_C in each time step $\Delta\tau$, otherwise the value of $\Delta\tau$ must be decreased. Then the time t is updated as $t+\Delta\tau$ and the simulation domain is switched again until the desired end of simulation. In the initial version of this algorithm, the transfer of molecules is based on whether or not they appear on the other side of the interface. The propensity per molecule for molecules to migrate from cubes adjacent to the interface I into Ω_M is the same as the propensity per molecule for molecules to jump from one cube to another. The procedure is summarized in Algorithm 4.

Algorithm 4 Splitting domain algorithm

(a) Initialize the number of molecules in the cubes in Ω_C and the positions of molecules that are in Ω_M at time $t = 0$.

(b) Choose $\Delta\tau$, the time between switching the simulation subdomains. Let $\Delta t = \frac{1}{10}\Delta\tau$, the time at which the next migratory event will take place in Ω_M.

(c) Freeze Ω_M and simulate Ω_C

- Add a layer of empty cubes adjacent to the interface I which contains n^2 same cubes in Ω_C and consider it to be part of Ω_C.
- Initialize all molecules which migrated from Ω_M to Ω_C in the previous time step.
- Change the number of molecules in Ω_C according to Algorithm 2 with the desired end of simulation $\Delta\tau$.

(d) Freeze Ω_C and simulate Ω_M

- Add a layer of empty cubes adjacent to the interface I and consider it as part of Ω_M.
- Initialize all molecules which migrated from Ω_C to Ω_M in the previous time step.
- Change the positions of all molecules in Ω_M according to equations (5)$\sim$(7) for 10 times.

(f) Update $t := t + \Delta\tau$

(g) Repeat steps **(c)**, **(d)** and **(f)** until the desired end of simulation.

5.2 Optimization of the initial approach

The initial approach of the algorithm gives the result with a jump over the interface. See Figure 8(a)$\sim$8(d). We introduce the following factors to improve the accuracy.

1 Use the coefficient Φ as stated in (19).

2 Use the probability P_m as stated in (20).

3 Sample the x-coordinates of molecules moving from Ω_C to Ω_M with help of the probability distribution $f(x)$ as stated in (17).

We also combine the above factors to check the convergence and analyze the errors.

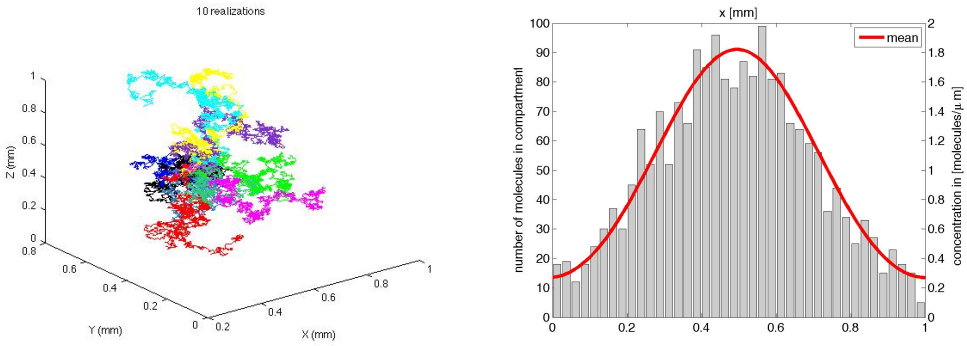
6 Experiments

In this section, the performance of the algorithms is discussed. All simulations have been done by MATLAB and we mainly focus on the numerical efficiency and the accuracy of our algorithms. The comparison between algorithms is also included.

6.1 Molecular-based modeling

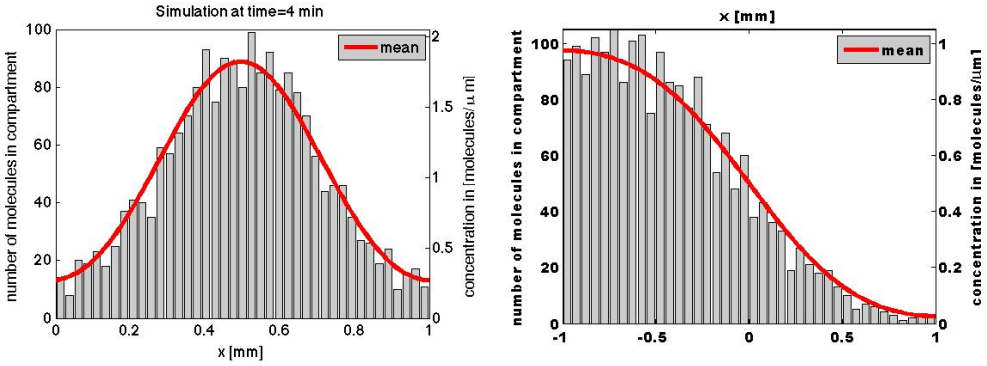
The Molecular-based algorithm, Algorithm 1, is implemeted here. Choosing $D = 10^{-4}mm^2s^{-1}$, $L = 1mm$, $X(0) = Y(0) = Z(0) = 0.5$ and $\Delta t = 0.1sec$, we plot ten realizations at time $t = 240s$. See Figure 6(a).

Convergence can be checked based on the law of large numbers, i.e. the stochastic simulation should converge to the deterministic counterpart with the assumption that the number of molecules



(a) (b)

Figure 6: Trajectories of molecules and the distribution with respect to x



(a) (b)

Figure 7: (a) Cube-based model (b)The two-regime method

is large enough. Assume that there are 2000 molecules in Ω, each of which follows the molecular-based algorithm in Algorithm 1. We use the same coefficient: $D = 10^{-4}mm^2s^{-1}$, $L = 1mm$, $X(0) = Y(0) = Z(0) = 0.5$, $\Delta t = 0.1s$ for all molecules and we simulate up to $240s$. We choose one of the three axes, say x, and divide the domain $[0, L]$ into $n = 40$ intervals, each of which has the length $h = \frac{L}{40}$. Then we calculate the number of molecules in each interval and plot them as histograms shown in Figure 6(b). The mean value computed in (4) is also plotted in the same figure for comparison. Good convergence is observed here.

6.2 Cube-based modeling

The Cube-based algorithm, Algprithm 2 is implemeted here. Choosing $D = 10^{-4}mm^2s^{-1}$, $L = 1mm$ and initial condition $A_0(21, 21, 21) = 2000$ with $n = 41$, we simulate up to $240s$. For comparison we do the same thing here as in molecular-based modeling. The number of moleclues in each interval with respect to x is plotted as a histogram in Figure 7(a). The mean value in (4) is also plotted in the same figure. Good convergence is observed.

6.3 Two-regime method from University of Oxford

The two-regime method described by Algorithm 3 is implemeted here. Choosing $\Delta t = 10^{-6}s$, $D = 1mm^2sec^{-1}$, $L = 1mm$, we simulate with 2000 molecules up to 0.1 s. In the beginning, all molecules are placed in Ω_C in cubes with equal probability. We plot the distribution of x and the

mean value together as shown in Figure 7(b). Good convergence is observed.

6.4 Splitting domain algorithm

Here we implement Algorithm 4. All the simulations are done with $L = 1mm$ and $n = 20$. Other coefficients are written in Table 1 and figures are shown in Figure 8 and Figure 9. There are 20 layers of cubes perpendicular to the x-axis in Ω_C.

In the beginning, all molecules can either start in the left part Ω_C and identical number of molecules is assigned to each cube, or start in the right part Ω_M and the exact positions are assigned to all molecules. In the latter case, we assume that there are also small cubes in Ω_M, with the same side length of the cubes in Ω_C. Identical number of molecules is assigned to each cube. The coordinates of the molecules are given with the assumption that they are in the midpoints of these cubes.

The explanations for the figures are as follows. Numbers $1 \sim 8$ correspond to figures $(a) \sim (h)$ in Figure 8 and numbers $9 \sim 10$ correspond to figures $(a) \sim (b)$ in Figure 9.

1 In Figure 8(a), we can see that it converges except the part near the interface. It indicates that the way we treat the molecules moving from Ω_C to Ω_M is not accurate. More molecules should move from Ω_C to Ω_M. Thus increasing the diffusion rate of those molecules may be a solution.

2 In order to see the jump near the interface clearly, we use a smaller time step. In Figure 8(b), we can see that the jump is obvious.

3 Here we use 10000 molecules. It converges quite well except the part near the interface.

4 When we simulate up to $2s$, the mean value tends to a horizontal line beacuse of the law of large numbers.

5 The coefficient Φ in (19) is introduced here. Figure 8(e) shows that there is no significant jump near the interface.

6 We use a smaller time step here. Good agreement is observed, but it seems that a little bit more molecules are in the right part.

7 The distribution function $f(x)$ in (17) is introduced here. With a high probability that x sampled from $f(x)$ is smaller than it would be if it was sampled from a uniform distribution. More molecules will move from Ω_M to Ω_C.

8 Simulation up to $2s$. Compare with Figure 8(d)

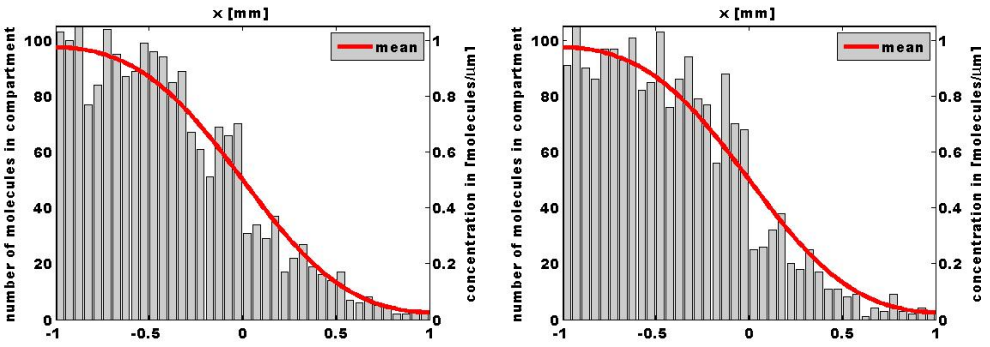
9 When the molecules start from the right part without any corrections, the jump near the interface is observed.

10 When the molecules start from the right part with the correction Φ in order to eliminate the jump, good convergence is observed. But there are still more molecules in the starting (right) part.

6.5 Comparison

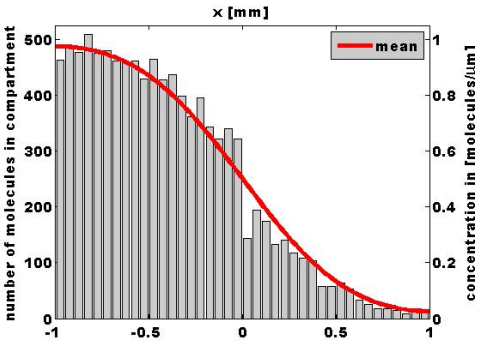
In this section, we compare the splitting domain method with corrections to the splitting domain method without corrections by computing the relative errors of Ω_C and Ω_M.

We simulate 2000 molecules starting from Ω_C with $L = 1mm$ and $n = 20$. Thus, there are 20 layers of cubes which are parallel to y-z-plane in Ω_C and each of which contains 100 molecules. The molecules are placed randomly in y and z directions. Firstly, we simulate using the splitting domain algorithm in Algorithm 4 without any corrections. The value of $\Delta\tau$ is chosen from

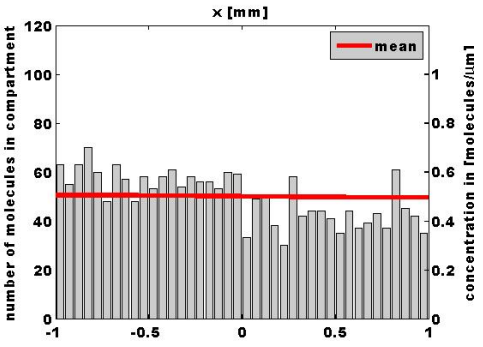


(a)

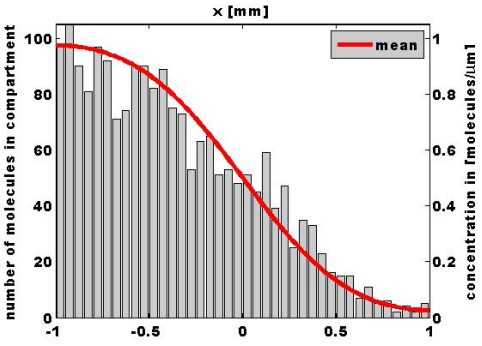
(b)



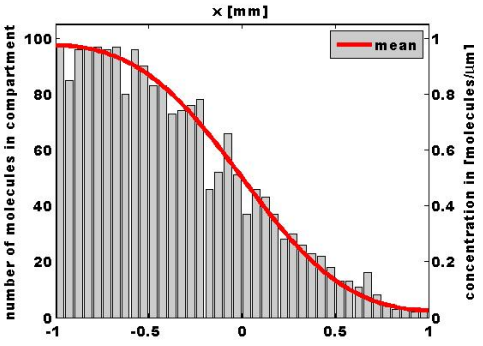
(c)



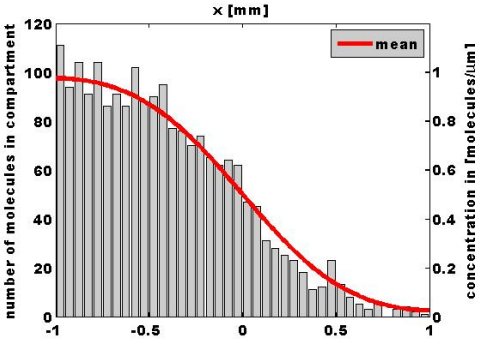
(d)



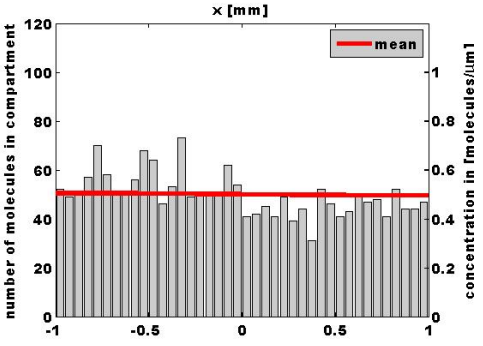
(e)



(f)



(g)

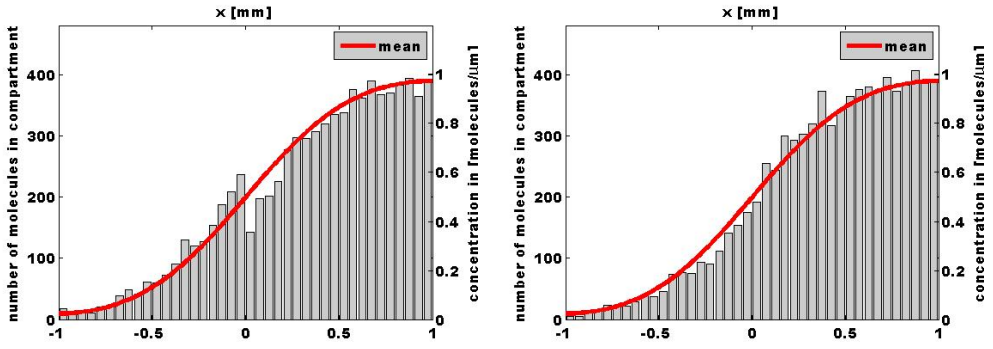


(h)

Figure 8: Splitting domain method, starting from Ω_C

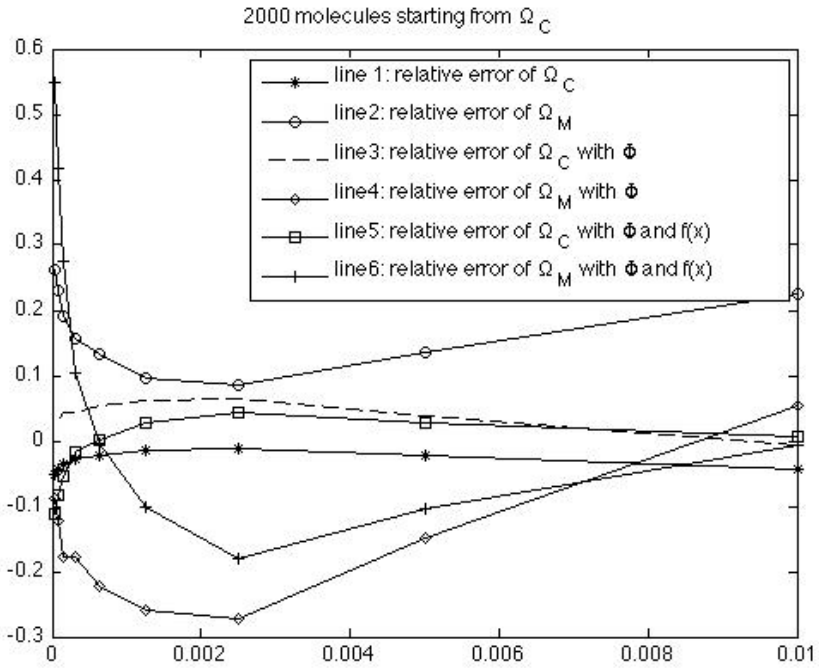
Table 1: Coefficients used in the splitting domain algorithm

Index	D/mm^2s^{-1}	$\Delta\tau/s$	*number of molecules*	desired time T/s	Correction
starting from Ω_C					
1	1	$\frac{1}{3200}$	2000	0.1	None
2	1	10^{-6}	2000	0.1	None
3	1	$\frac{1}{3200}$	10000	0.1	None
4	1	$\frac{1}{3200}$	2000	2	None
5	1	$\frac{1}{3200}$	2000	0.1	Φ
6	1	10^{-6}	2000	0.1	Φ
7	1	$\frac{1}{3200}$	2000	0.1	Φ and $f(x)$
8	1	$\frac{1}{3200}$	2000	2	Φ and $f(x)$
starting from Ω_M					
9	1	$\frac{1}{3200}$	8000	0.1	None
10	1	$\frac{1}{3200}$	8000	0.1	Φ

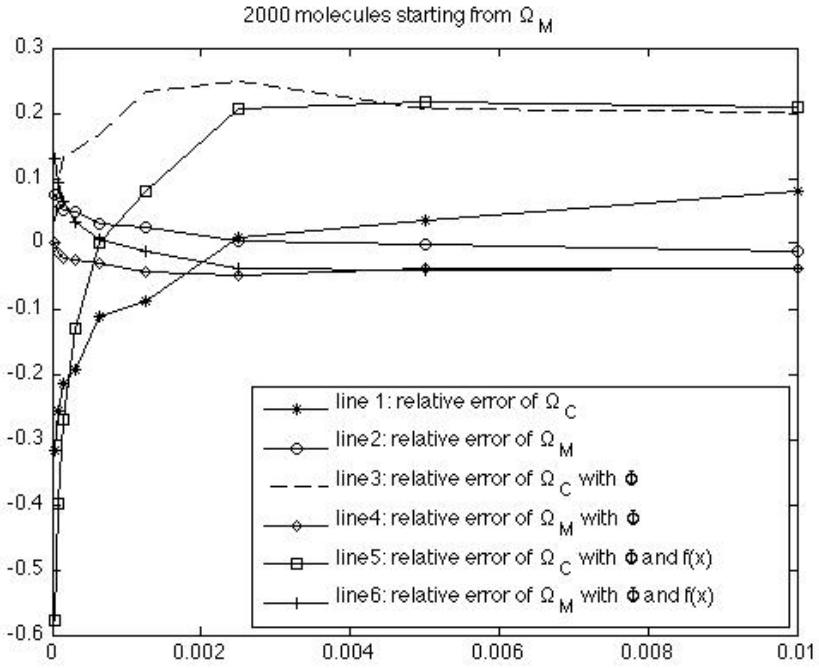


(a) (b)

Figure 9: Splitting domain method, starting from Ω_M



(a)



(b)

Figure 10: (a)relative error starting from Ω_C (b)relative error starting from Ω_M

$[\frac{1}{25600}, \frac{1}{12800}, \frac{1}{6400}, \frac{1}{3200}, \frac{1}{1600}, \frac{1}{800}, \frac{1}{400}, \frac{1}{200}, \frac{1}{100}]$ one by one. As the algorithm is stochastic, the simulation is repeated for five times for each $\Delta\tau$. Meanwhile, the relative errors for both Ω_C and Ω_M are calculated at time $t = 0.1s$ by

$$\eta = \frac{I - \Sigma}{I} \tag{21}$$

where I is the mean value and Σ is the number of molecules in each compartment. We do not use the absolute value because the sign indicates which subdomain has more molecules. The averages of the relative errors are plotted in Figure 10(a). Secondly, we simulate using the same algorithm with the correction Φ in (19). Finally, the probability function $f(x)$ in (17) is included in simulation. The relative errors for both Ω_C and Ω_M at time $t = 0.1s$ are plotted in the same figure for comparison. In this figure, we can see that the relative errors are dependent on $\Delta\tau$. Line 1, 3 and 5 have the same shape. Line 1 is below 0. It indicates that more molecules should move from Ω_C to Ω_M. When Φ is included, we get Line 3 which is above 0. It indicates that too many molecules have moved from Ω_C to Ω_M. When $f(x)$ is used to give the initial positions of molecules moving from Ω_C to Ω_M, we get Line 5 which is between Line 1 and Line 3. An important issue to be considered is that Line 5 goes across 0 at a certain value of $\Delta\tau$. If we use this value in the simulation, we can get a result that the stochastic simulation converges to the deterministic counterpart with relative error 0. The same conclusion can be made based on Line 2, 4 and 6.

We redo the experiment above but in the beginning, initilize uniformly all molecules in Ω_M. The relative errors are plotted in Figure 10(b). The same conclusion as in Figure 10(a) can be made.

7 Conclusion

The stochastic simulations in the molecular-based model and cube-based model converge to the deterministic counterpart obtained by the analytical solution of the corresponding partial differential equations. The TRM algorithm is implemented in 3D and good agreement is observed. However, the present implementation is not computationally efficient. The splitting domain algorithm suffers from the convergence problem. When the algorithm is implemented straightforeardly, these occurs a jump near the interface. Several coefficients are introduced to improve accuracy. Better convergence is observed in the figures. The future work could be done by computing mathematically the corrections for molecules moving across the interface and determining the value of $\Delta\tau$ which gives zero relative errors.

Acknowledgements

I would like to thank my supervisor Per Lötstedt for his guidance of the project, valuable time for discussion and constructive suggestions of the report and poster. I am grateful to Stefan Hellander for his detailed answers to my questions. I thank Maya Neytcheva for her help with LaTeX and the report.

References

[1] S.S. Andrews 2009, Accurate particle-based simulation of adsorption, desorption and partial transmission. *Phys. Biol.* **6**, 046015. (doi:10.1088/1478-3975/6/4/046015)

[2] S.S. Andrews, D. Bray 2004, Stochastic simulation of chemical reactions with spatial resolution and single molecule detail *Phys. Biol.* **1**, 137-151. (doi:10.1088/1478-3967/1/3/001)

[3] H. Berg 1983, Random Walks in Biology. Princeton University Press.

[4] R. Erban, S.J. Chapman 2007, Reactive boundary conditions for stochastic simulations of reaction-diffusion processes. *Phys Biol* 4(1), 16-28

[5] R. Erban, S.J. Chapman and P.K. Maini 2007, A practical guide to stochastic simulations of reaction-diffusion processed. *arXiv*: 0704.1908v2 [q-bio.SC]. `http://people.maths.ox.ac.uk/~erban/Education/StochReacDiff.pdf`

[6] M.B. Flegg, S.J. Chapman and R. Erban 2011, The two-regime method for optimizing stochastic reaction-diffusion simulations *J. R. Soc. Interface* (doi:10.1098/rsif.2011.0574)

[7] M.A. Gibson and J. Bruck 1999, Efficient exact stochastic simulation of chemical systems with many species and many channels *J. Phys. Chem. A* 2000, 104, 1876-1889

[8] D.T. Gillespie 1976, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions *J. Comput. Phys.*, 22(4):403-434

[9] J. Elf and M. Ehrenberg 2004, Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases *Syst. Biol, Vol. 1, No. 2* (doi: 10.1049/sb:20045021)

[10] A. Hellander, S. Hellander and P. Lötstedt 2011, Coupled mesoscopic and microscopic simulation of stochastic reaction-diffusion processes in mixed dimensions. *Technical report/Department of Information Technology, Uppsala University, ISSN: 1404-3203; 2011-005*

[11] Wikipedia, Fick's laws of diffusion, `http://en.wikipedia.org/wiki/Fick's_laws_of_diffusion`