Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Improved two-regime method for spatial stochastic simulations of cellular signaling

Adrian Martinez Gomez

October 6, 2013

Semester project

Abstract

Signaling in the cell transmits external signals received at the surface by receptors to the nucleus inside of the cell, where respective genes are regulated in response to the signal. In addition to the spatial component also noise due to low particle number has to be included in the simulation. Two simulation regimes are well-used for such simulations, namely (1) particle-based simulations that track every molecule on its diffusive path through the cell and (2) compartment-based simulations that only keep track of the number of molecules in sub-compartments defined by a spatial discretization of the cell volume. While particle-based simulations have a higher accuracy and resolution of the process than compartment-based methods, they are also computationally more demanding. Therefore Two- Regime Methods have been developed to combine the resolution of the particle-based regime with the performance advantage of the compartment-based method. The project investigated improved methods to couple both regimes, especially by increasing the inherent spatial resolution of the compartment-based simulation method. This improved the boundary conditions between the two regimes, and smoothened the effect of the interface compared to the original model.

Contents

1 Introduction

Originally, the field of signaling biology has suffered the lack of a well-established modeling and communication language [7]. The quantitative tools of systems biology used in the last years towards a better description of simulation of space and time [5] have proven to be a big step towards a more precise analysis.

Molecular-based models describe the spatial-temporal dynamics of biological systems. And, although these models provide a higher level of detail, they are computationally expensive. Therefore, compartment-based models, as the one introduced by Gillespie [4], are used to preserve computational power in systems with high abundance of particles.

In the multiscale environment of cellular signaling, hybrid methods can be employed to optimize the use of this computational resources [6], while still mantaining a high accuracy. This kind of methods are sometimes referred as two-regime methods (TRM), and have been developed recently for optimizing stochastic reaction-diffusion simulations [2]. The hybrid model usually consists of a spatial continuous - and a spatial discrete lattice-based model. However, this hybrid methods contain core problems, namely the partition of the simulation domain into regimes and the definition of a correct interface between these two regimes.

The hybrid method presented here developes the model presented by Klann et al. [6] and partially by Flegg et al. [2], which combine Brownian dynamics particle tracking and Gillespie method, a step further.

Flegg et al. [2] do a clean derivation of the boundary conditions for the interface. However only a static boundary was regarded, and still an artificially increased diffusion is observed along the boundaries in simulations.

Klann et al. [6] use a more versatile approach, which works for more general cases. The method presented supports coupling of molecule species in reactions where one of them is in particle - and the other in compartment mode, as well as dynamic switching of domains. That model also has some issues, as for example the naive implementation of the boundary, which leads to artificial flux into the compartment domain.

But the relative simplicity of the Gillespie domain of the method demands further investigation towards a more accurate model description. An adaptive mesh size is introduced in this project to achieve this. An average offset for the particles in every voxel of the mesh is defined.

Reactions are the key player of signaling, and therefore a crucial part of the simulation [1]. For testing the refined TRM they can however be neglected, leaving diffusion as the only particle interaction. Possible implementations of reactions in the TRM will be outlined in the discussion on section §6. As a consequence of this we will assume there is only one molecular type.

Therefore, the objectives of this project can be divided into the following:

- **Developing an improved two-regime method for cellular signaling with adaptive mesh size.**

- **Comparing this improved method to the previous existing model developed by Klann et al. [6].**

The comparison between the methods will be done in terms of different concepts: accuracy, i.e. the error to the expected distribution, minimization of a bias to one of the domains, and computational time. A loss in computational performance is expected when doing the improvement in the two-regime method, however, a reasonable trade-off between this performance and the improvement in accuracy should be discussed.

This project is going to be developed by parts, which built into each other result in the final improved method. First, the Gillespie method will be implemented and tested for different particle numbers as well as voxel quantities. Then, an analysis of the particle based Brownian dynamics will be performed by making two experiments. First, the diffusion is tested, to verify the expected theoretical distribution through experimentation . Secondly, the offset is tested for a small domain to see if the particles redistribute themselves over time.

The hybrid method, which couples the two previous algorithms, is addressed later on.

Finally, the improved hybrid method will be implemented and compared to the naive hybrid method.

2 Gillespie algorithm

2.1 Theory

The main idea of the Gillespie algorithm is to divide the space Ω_C into U compartments $V_1, \ldots, V_U$ called voxels. Every one of these voxels then contains a number of particles. For a given voxel ν the number of particles is denoted as N_ν.

Diffusion with diffusion coefficient D_i, from a voxel ν to and adjacent voxel κ is a first-order transport reaction, and its propensity is given by

$$\alpha\left(N_\nu\right) = k_{\text{jump}}^{\nu\to\kappa} N_\nu. \tag{1}$$

The jumping rate constant

$$k_{\text{jump}}^{\nu\to\kappa} = \frac{D_i}{h^2} = \frac{D_i \cdot h^2}{h \cdot h^3} = \frac{D_i \cdot S_{\nu,\kappa}}{h \cdot V_{\nu,\kappa}} \tag{2}$$

depends on the voxel side length h, and the diffusion coeffiecient. (2) is a generalization for higher dimensions, $S_{\nu,\kappa}$ being the surface are of the voxel interfaces, and $V_{\nu,\kappa}$ the voxel volume.

According to Gillespie [4], the next diffusion event in voxel ν is given by the putative time

$$t_\nu = t + \varepsilon_\nu. \tag{3}$$

In (3), t is the current simulation time and ε is an exponential distributed random number of mean $\mu_{\varepsilon_\nu} = \frac{1}{\alpha(N_\nu)}$. This time has to be recomputed every time a diffusion event takes place.

2.2 Implementation

The Gillespie algorithm used in the computations is as follows

Algorithm 1 Gillespie Algorithm

Require: Ω_C, t, t_{End}, $\triangle t$, $[N_1, \ldots, N_U]$

1: **while** $t < t_{\text{End}}$ **do**
2: $[t', \nu] = \texttt{getFirstDiffusionTime}() = \text{minloc}\left([t_1, \ldots, t_U]\right)$
3: $t = t'$
4: $N_\nu = N_\nu - 1$
5: $t_\nu = t + \varepsilon_\nu$
6: $\kappa = \texttt{selectNewVoxel}()$ *(New voxel has to be adjacent)*
7: $N_\kappa = N_\kappa + 1$
8: $t_\kappa = t + \varepsilon_\kappa$
9: **end while**

Note that voxels only connect to neighboring voxels which can accept particles and are inside of the simulation domain . The function `selectNewVoxel`() therefore only returns valid new voxels κ.

2.3 Results

To be able to do a valid simulation of the Gillespie Algorithm on the preceding page, the following setting for a one dimensional problem is simulated.

All the particles start from a delta distribution at the value $x_0 = 60$. This eases up some functions in the implementation, for example the `selectNewVoxel`() one.

The random uniform distribution is simulated, where for a domain $\Omega_C = [1.0, 10.0]$ with voxel length $h = 1$, $\mathbf{N}_{\text{particles}} = 10000$ were generated with the pseudorandom uniform distribution function `rand`(). The result is depicted in figure 1, where the simulation was run from $t = 0$ s to $t_{\text{End}} = 20$ s, with $\triangle t = 0.01$s.

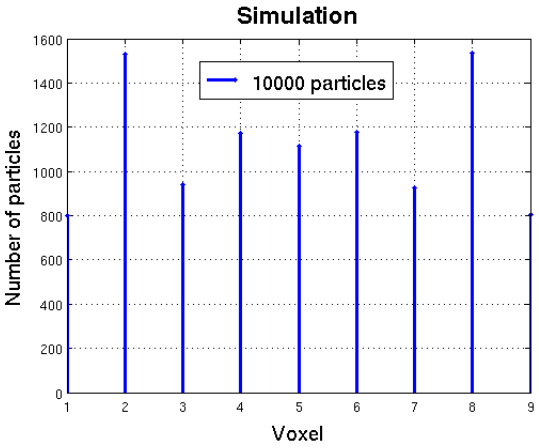

Figure 1: Gillespie simulation for $\mathbf{N}_{\text{particles}} = 10000$.

A huge benefit of Gillespie is the low running time $t_{runtime}$, which averaged for 10 simulation runs was found to be 0.7578 s.

3 Particle algorithm

3.1 Theory of the Brownian dynamics particle tracking

Trajectories of large biomolecules (e.g. proteins) are computed through Brownian dynamics [8]. In time-driven algorithms, the diffusion of the j-th particle is computed as follows with the Brownian dynamics:

$$\mathbf{x}_j\left(t+\triangle t\right)=\mathbf{x}_j\left(t\right)+\triangle\mathbf{x}\cdot\xi \tag{4}$$

with ξ being a three-dimensional zero-mean Gaussian random vector of unit variance, and $\triangle\mathbf{x}=\sqrt{2D_j\triangle t}$ being the moved distance. This is a discretization of the Brownian motion

$$\mathrm{d}\mathbf{X}_j=\sqrt{2D_j}\mathrm{d}\mathbf{W}_j \tag{5}$$

Particles can not step into cellular structures as for example plasma membrane, nucleus, cytoskeleton, etc. [6]. Corresponding steps $\mathbf{x}_j\left(t+\triangle t\right)$ in forbidden domain, e.g. that try to escape Ω_M, are reset to the valid domain: $\mathbf{x}_j\left(t+\triangle t\right)=\mathbf{x}_j\left(t\right)$.

The expected theoretical distribution is

$$p\left(\mathbf{x},t\right)=\frac{1}{\sqrt{4\pi Dt}}\exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}-\mathbf{x}_0}{\sqrt{2Dt}}\right)^2\right) \tag{6}$$

for a delta distribution, where at the beginning all the particles are located at position $\mathbf{x}_0$.

Since reactions are going to be neglected, no further considerations are needed for testing the diffusion driven events.

3.2 Implementation and Validation

This Brownian model may be easy to implement and test, but it is not suited for models with high particle concentrations, as the computational time is expected to increase linearly.

In this section we want to test the diffusion experiment, where particles are simulated for the Brownian dynamics. The particle domain $\Omega_M=(-\infty,\infty)$ is assumed, and $\mathbf{N}_{\text{particles}}=10000$ particles are initializated with a delta distribution at $x_0=0.0$.

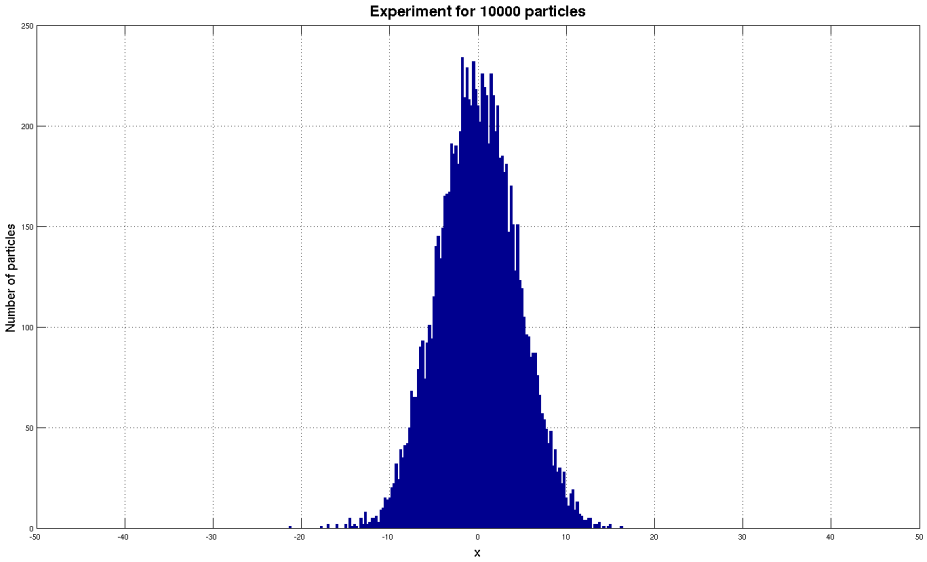

Figure 2: Obtained particle distribution

The simulation ran from $t_0 = 0.0$ s to $t_{\text{End}} = 10.0$ s, with time intervals of $\triangle t = 0.01$ s.

Figure 2 shows the particle distribution obtained at the final time t_{End}. As can be seen, the particles tend to diffuse from their starting position x_0.

The averaged mean observed expected value for 10 trials is $\mu_{\text{obs}} = 0.0602$, which is close to the expected mean $\mu_{\text{expected}} = 0.0$. Also, the averaged observed standard deviation for 10 trials is $\sigma_{\text{obs}} = 4.4346$. Of course, the standard deviation increases with t_{End}, as particles diffuse for longer time. The expected sigma is $\sigma_{\text{expected}} = \sqrt{2Dt_{\text{End}}} = 4.472$, showing that the particle method is implemented correctly.

Figure 3 shows the comparison to the expected normal distribution from equation (6).

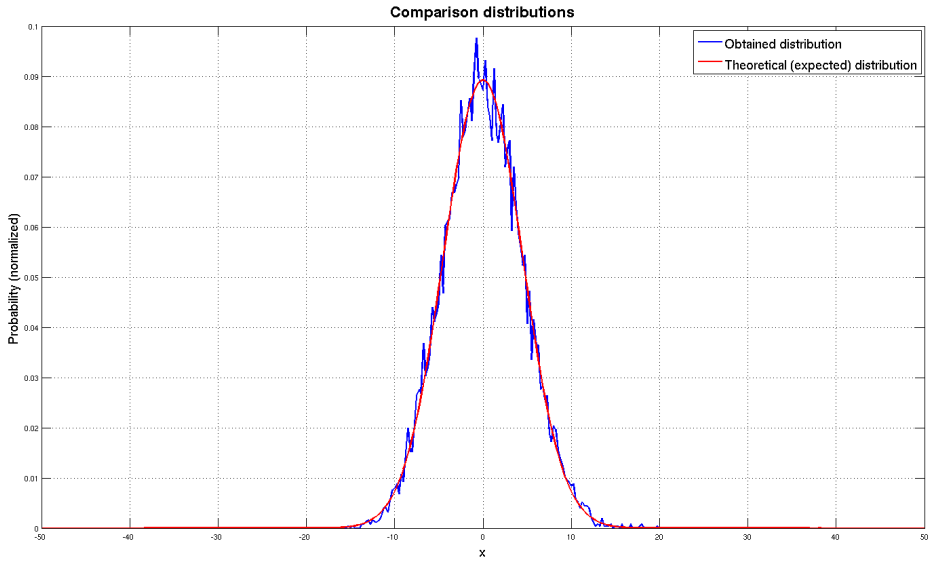

Figure 3: Normalized distribution (6) validated

It is important to note that for this $\mathbf{N}_{\text{particles}} = 10000$ particles the runtime was $t_{\text{runtime}} = 37.4$ s, whereas for $\mathbf{N}_{\text{particles}} = 1000$ particles $t_{\text{runtime}} = 3.76$. This shows the linear increase in the runtime t_{runtime} with respect to $\mathbf{N}_{\text{particles}}$. Due to the random nature of the process, the computed average properties become more exact with more particles, however at the cost of increased computation time.

On the other hand, simulating also $\mathbf{N}_{\text{particles}} = 10000$ with the Gillespie algorithm from section §2, the runtime was significantly lower: $t_{\text{runtime}} = 1.038$ s.

3.3 Offset particle-based simulations in a bounded domain

Next, a new experiment is defined to test the offset. We are given a Particle domain $\Omega_M = [0.0, 1.0]$ and $\mathbf{N}_{\text{particles}} = 10000$ particles distributed with a delta distribution at different starting positions $\mathbf{x}_0 = \{0.6, 0.7, 0.8, 0.9\}$. The aim with this simulation is to extract the behavior of the offset of the particle mean of the center of the voxel for a voxel of length $h = 1$, which is the length of Ω_M here, and try to match this decaying offset with a first order kinetics function.

Note that starting positions $\mathbf{x}_0$ smaller than $x_0^* = 0.5$ are not considered because of redundancy with the selected ones.

The exponential function

$$f(t) = (x_0 - x_0^*) \cdot e^{-c_2 \cdot t} \tag{7}$$

is fitted to the decaying offset, where $x_0 \in \mathbf{x}_0$ and c_2 is the decaying constant.

The setting is simulated from $t = 0$ s to $t_{\text{End}} = 1$ s, with time intervals $\triangle t = 0.01$ s.

As can be seen in figure 4 all the offsets decay to zero fast. Function (7) with the decaying constant $c_2 = 8.5$ fits this decay well.

This value will be stored for further use in section 5.2.

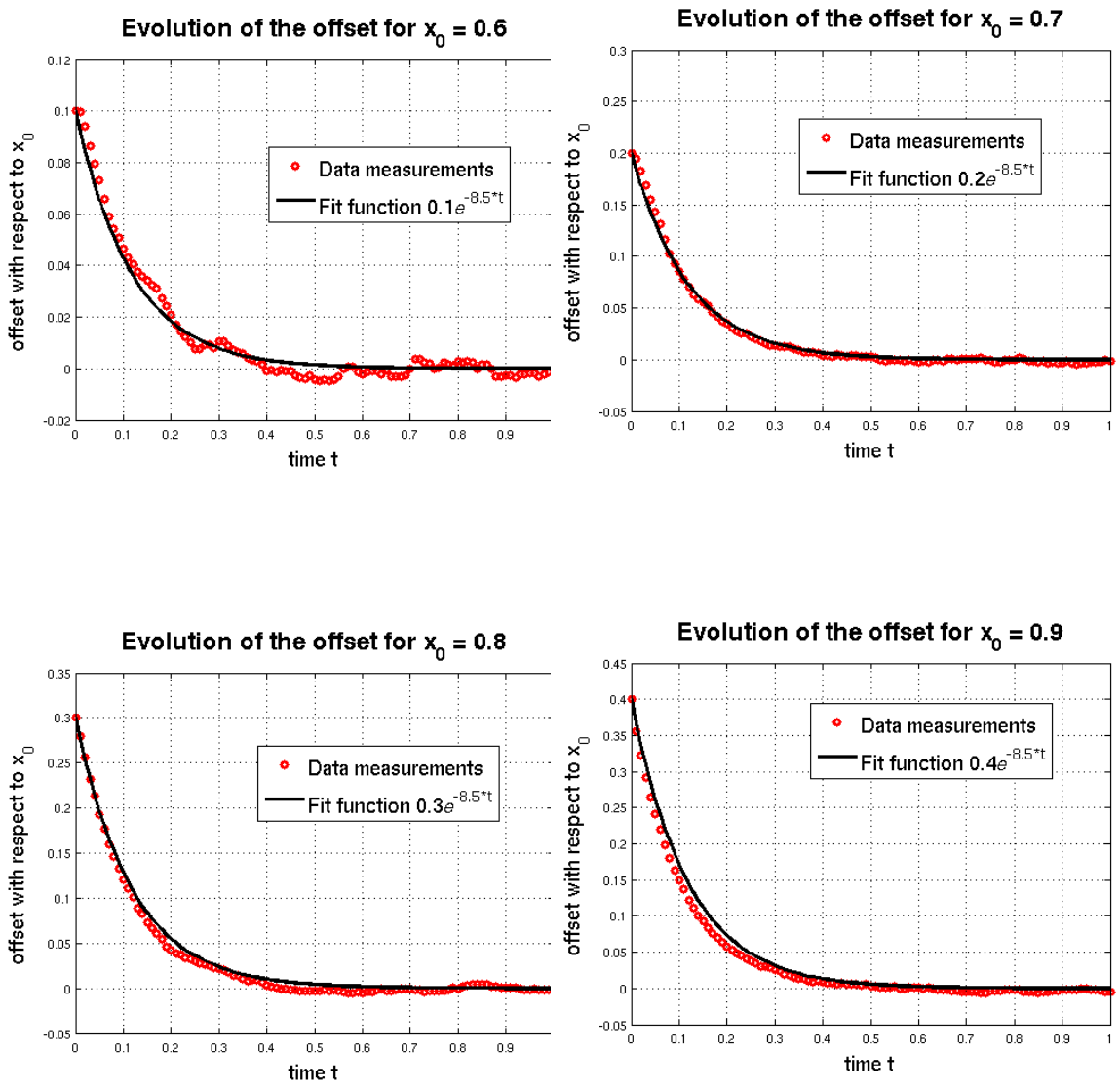

Figure 4: Particle simulation for different initial offsets $\mathbf{x}_0 = \{0.6, 0.7, 0.8, 0.9\}$

4 Hybrid method

4.1 General idea and Algorithm

As can be seen on the scheme in figure 5, the domain is split into two regions: Ω_c is the Gillespie region, where voxels of size h containing number of particles $[N_1, \ldots, N_U]$ are defined, whereas Ω_M is the space where the particles move according to the Brownian dynamics explained in section §3.

For diffusion out of voxel ν into the new voxel κ, it has to be tested if κ is also in Ω_C. Otherwise, a particle has to be activated in the particle domain Ω_M, and it adquires a position $x_j(t)$ from a uniform distribution in voxel κ.

Likewise, if a particle diffuses into Ω_C, then it has to be destroyed and the number of molecules in the corresponding voxel is increased by one. Without reactions, the sum of all the particles plus the concentrations in the voxels of the Gillespie domain remains constant.

A key feature of this hybrid algorithm are the putative times. Some useful notation of Flegg et al. [2] was taken. t_C denotes the time for the next Gillespie C-event, whereas t_M is the time until the next Particle M-event.

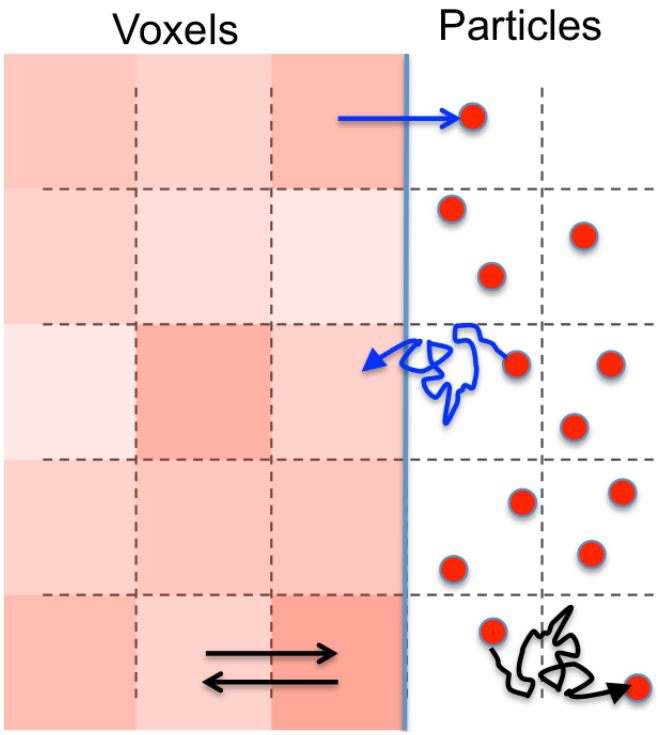

Figure 5: Schema of the hybrid method

The hybrid algorithm is shown here. The implementation was taken from Klann et al. [6], with some slight notation adapted from Flegg et al. [2, 3] for simplicity.

Algorithm 2 Hybrid Algorithm

Require: $\Omega_C, \Omega_M, t, t_{End}, \triangle t, [t_1, \ldots, t_U], [N_1, \ldots, N_U], \mathbf{x}$

$t_M = t$
$t_C = \min([t_1, \ldots, t_U])$
$\triangle x = \sqrt{2D\triangle t}$
while $t < t_{End}$ **do**
  if $t_C < t_M$ *(Gillespie event occurs)* **then**
    $t = t_C$
    $[\sim, \nu] =$ `getFirstDiffusionTime`$() = \text{minloc}([t_1, \ldots, t_U])$
    $N_\nu = N_\nu - 1$
    $t_\nu = t + \varepsilon_\nu$
    $\kappa =$ `selectNewVoxel`$()$ *(New voxel has to be adjacent)*
    if new voxel κ in Ω_M **then**
      `createParticle`$()$
    else
      $N_\kappa = N_\kappa + 1$
      $t_\kappa = t + \varepsilon_\kappa$
    end if
    $t_C = \text{minloc}([t_1, \ldots, t_U])$
  else if $t_M < t_C$ *(Particle event occurs)* **then**
    $t = t_M$
    for all particles **do**
      if particle j in Ω_M **then**
        $x_j(t + \triangle t) = x_j(t) + \triangle x \cdot \xi$
        if $x_{j,new}$ in forbidden region **then**
          $x_j(t + \triangle t) = x_j(t)$
        else if $x_j(t + \triangle t)$ jumps into voxel κ in Ω_C **then**
          `deleteParticle`(j)
          $N_\kappa = N_\kappa + 1$
          $t_\kappa = t + \varepsilon_\kappa$
        end if
      end if
    end for
    $t_M = t_M + \triangle t$
    $t_C = \text{minloc}([t_1, \ldots, t_U])$
  end if
end while

4.2 Results

A simulation for $\mathbf{N}_{\text{particles}} = 10000$ particles starting with a random standard uniform distribution can be seen in figure 6. The space is split into Gillespie- $\Omega_C = [1.0, 31.0]$ and particle- $\Omega_M = [31.0, 41.0]$ domains, thus the interface is at $I = 31.0$.
The simulation runs from $t = 0$ s to $t_{\text{End}} = 10$ s, with time intervals $\triangle t = 0.01$ s. The average of 10 run results were made for better statistics.

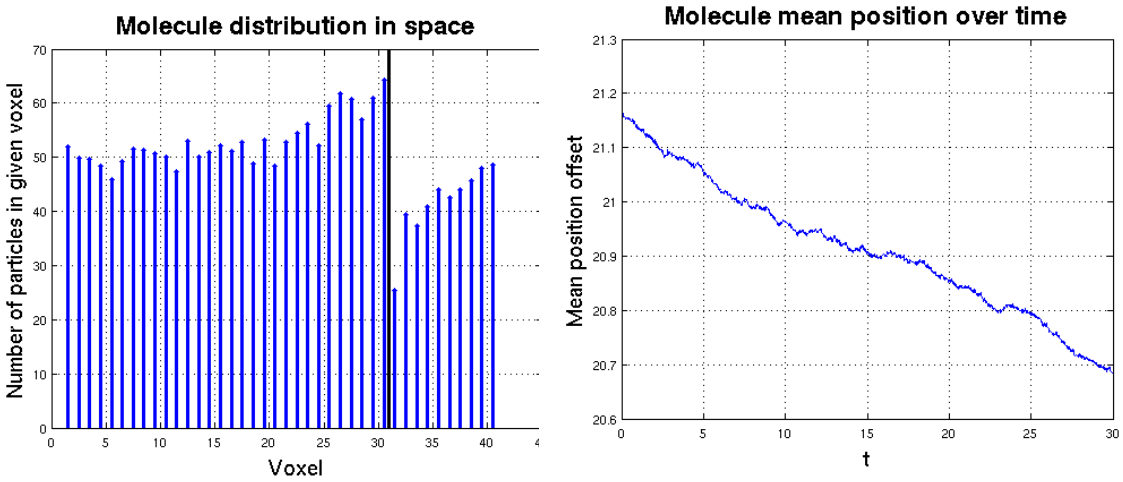

Figure 6: Simulation of uniform distribution

As is made clear from the distribution in figure 6, the mean offset decays too fast, and molecules tend to move to the left (the Gillespie domain Ω_C). This is an undesired particle flux due to the interface I.

Exactly this problem sets the motivation to develop an improved method to overcome this undesired flux.

5 Improved hybrid method

5.1 Introduction

The results obtained in section 4.2 are versatile compared to the ones from Flegg et al. [2, 3], since the model could admitt more than one molecular species, as well as reactions [6].

Flegg et al. [3] uses an interesting method, where the change in the propensity of migration Φ is defined and used. Also, a probability $\Psi \in [0, 1]$ is defined for transfering particles from Ω_M to Ω_C.

On contrary, the treatment of interfaces between regimes done with the model of section 4.2 are incomplete, and need further improvement. This is exactly what will be done in this section with the virtual adaptive mesh size.

5.2 The virtual adaptive mesh size

5.2.1 Algorithm

A new variable h', which is the virtual voxel size, is introduced in the algorithm. This enables for a more detailed description of the particle number and their position in the compartment based Gillespie domain, since an average offset to the voxel center is stored. This offset gives a more precise jumping propensity, because the particles are no longer assumed to be well-mixed, i.e. uniformly distributed with mean position in the center of the voxel.

We assume the offset decays to zero with first order kinetics

$$v_{\text{offset}}(\kappa) = v_{\text{offset}}(\kappa) \cdot e^{-k_{\text{offset}} \cdot (t - t_{\kappa,\text{offset}})}, \tag{8}$$

where $t_{\kappa,\text{offset}}$ is the offset time. This time has to be reset every time a change is made in voxel κ.

k_{offset} is the offset decay rate constant adjusted to run relaxation simulation tests, and is defined, similarly as in equation (2), as

$$k_{\text{offset}} = \frac{D}{h^2} \cdot c_2, \tag{9}$$

c_2 being the decaying constant found in section 3.2.

The jump of a new particle into a voxel κ has as a consequence that the voxel offset $v_{\text{offset}}(\kappa)$ has to be updated with the addition of the particle's offset in a weighted manner,

$$v_{\text{offset}}(\kappa) = \frac{N_\kappa}{N_\kappa + 1} \cdot v_{\text{offset}}(\kappa) + \frac{1}{N_\kappa + 1} \cdot p_{\text{offset}} \tag{10}$$

Thus, the virtual voxel size, which takes into account the weighted voxel offset, is defined in 1D as

$$h'(\kappa) = h - 2 \cdot v_{\text{offset}}(\kappa) \tag{11}$$

for every voxel κ. In figure 7 the virtual voxel size $h'(j)$ is depicted for 1D.

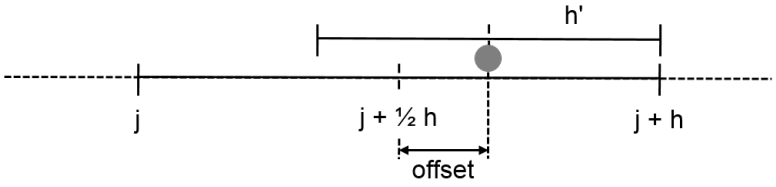

Figure 7: Virtual voxel size for 1D

Important to notice is also the change in the function `createParticle`(), which creates a particle if the jumps is from Ω_C to Ω_M. Now, the virtual voxel size has to be considered, and there is a subtle difference depending on whether the jump is to the right, in which case the formula is

$$x_j(t) = \kappa + h + \min(h', h) \cdot \xi, \tag{12}$$

or if the jump is to the left

$$x_j(t) = \kappa - \min(h', h) \cdot \xi, \tag{13}$$

where ξ is a normal distributed random number.

In Algorithm 3, the improved method is showed. Algorithms 4 and 5, helping functions for the improved algorithm are listed.

Algorithm 4 does the necessary changes for when a particle jumps out of a voxel, whereas Algorithm 5 regards the case that a particle jumps into the voxel.

Algorithm 3 Improved hybrid Algorithm

Require: $\Omega_C, \Omega_M, t, t_{\text{End}}, \triangle t, [t_1, \ldots, t_U], [N_1, \ldots, N_U], \mathbf{x}$

$t_M = t$
$t_C = \min([t_1, \ldots, t_U])$
$\triangle x = \sqrt{2D\triangle t}$
while $t < t_{\text{End}}$ **do**
  if $t_C < t_M$ *(Gillespie event occurs)* **then**
    $t = t_C$
    $[\sim, \nu] =$ `getFirstDiffusionTime`$() = \text{minloc}([t_1, \ldots, t_U])$ * `GillespieJumpOut` ()
    $\kappa =$ `selectNewVoxel`() *(New voxel has to be adjacent)*
    $p_{\text{offset}} = -p_{\text{offset}}$
    if jump to the right **then**
      $p_{\text{offset}} = \min(p_{\text{offset}}, 0.0)$
    else
      $p_{\text{offset}} = \max(p_{\text{offset}}, 0.0)$
    end if
    if new voxel κ in Ω_M **then**
      $h' = h - 2 \cdot p_{\text{offset}}$
      `createParticle`() *(Depending on jumping direction value differs)*
    else
      `GillespieJumpIn` ()
    end if
    $t_C = \text{minloc}([t_1, \ldots, t_U])$
  else if $t_M < t_C$ *(Particle event occurs)* **then**
    $t = t_M$
    for all particles **do**
      if particle j in Ω_M **then**
        $x_j(t + \triangle t) = x_j(t) + \triangle x \cdot \xi$
        if $x_{j,\text{new}}$ in forbidden region **then**
          $x_j(t + \triangle t) = x_j(t)$
        else if $x_j(t + \triangle t)$ jumps into voxel κ in Ω_C **then**
          `deleteParticle`(j)
          `GillespieJumpIn` ()
        end if
      end if
    end for
    $t_M = t_M + \triangle t$
    $t_C = \text{minloc}([t_1, \ldots, t_U])$
  end if
end while

Algorithm 4 Function `GillespieJumpOut` ()

$N_\nu = N_\nu - 1$
if $k_{\text{offset}} > 0$ **then**
  $v_{\text{offset}}(\kappa) = v_{\text{offset}}(\kappa) \cdot e^{-k_{\text{offset}} \cdot (t - t_{\text{offset}})}$
  $t_{\text{offset}} = t$
  $p_{\text{offset}} = v_{\text{offset}}(\kappa)$
else
  $v_{\text{offset}}(\kappa) = 0.0$
  $p_{\text{offset}} = 0.0$
end if
$h'(\kappa) = h - 2 \cdot v_{\text{offset}}(\kappa)$
if $h'(\kappa) < h_{\min}$ **then**
  $h'(\kappa) = h_{\min}$
else
  if $h'(\kappa) > 2 - h_{\min}$ **then**
    $h'(\kappa) = 2 - h_{\min}$
  end if
end if
$k_{\text{jump,left}} = \frac{D \cdot \texttt{isConnection}(\kappa, \texttt{left})}{(2h - h')^2}$
$k_{\text{jump,right}} = \frac{D \cdot \texttt{isConnection}(\kappa, \texttt{right})}{(h')^2}$
$t_\nu = t + (k_{\text{jump,left}} + k_{\text{jump,right}}) \cdot \varepsilon_\nu$

Algorithm 5 Function `GillespieJumpIn` ()

1: $p_{\text{offset}} = x_j\,(t + \triangle t) - \left(\kappa + \frac{h}{2}\right)$
2: **if** $k_{\text{offset}} > 0$ **then**
3: $\quad v_{\text{offset}}(\kappa) = v_{\text{offset}}(\kappa) \cdot e^{-k_{\text{offset}} \cdot (t - t_{\text{offset}})}$
4: **else**
5: $\quad v_{\text{offset}}(\kappa) = 0.0$
6: $\quad p_{\text{offset}} = 0.0$
7: **end if**
8: $v_{\text{offset}}(\kappa) = \frac{N_\kappa}{N_\kappa + 1} \cdot v_{\text{offset}}(\kappa) + \frac{1}{N_\kappa + 1} \cdot p_{\text{offset}}$
9: $t_{\text{offset}} = t$
10: $N_\kappa = N_\kappa + 1$
11: $h'\,(\kappa) = h - 2 \cdot v_{\text{offset}}(\kappa)$
12: **if** $h'\,(\kappa) < h_{\min}$ **then**
13: $\quad h'\,(\kappa) = h_{\min}$
14: **else**
15: $\quad$ **if** $h'\,(\kappa) > 2 - h_{\min}$ **then**
16: $\quad\quad h'\,(\kappa) = 2 - h_{\min}$
17: $\quad$ **end if**
18: **end if**
19: $k_{\text{jump,left}} = \frac{D \cdot \texttt{isConnection}(\kappa, \texttt{left})}{(2h - h')^2}$
20: $k_{\text{jump,right}} = \frac{D \cdot \texttt{isConnection}(\kappa, \texttt{right})}{(h')^2}$
21: $t_\kappa = t + \left(k_{\text{jump,left}} + k_{\text{jump,right}}\right) \cdot \varepsilon_\kappa$

5.2.2 Offset test

As seen in the offset test in section 3.2, a decaying constant $c_2 = 8.5$ was found. This modifies the value for the offset jumping rate constant, which now is

$$k_{\text{offset}} = \frac{D}{h^2} \cdot c_2 = \frac{1.0}{1.0^2} \cdot 8.5 = 8.5. \tag{14}$$

Next, a simulation is shown to see the importance of this value. Assume that $\mathbf{N}_{\text{particles}} = 2000$ are initiated with a delta distribution at $x_0 = 13.0$ in a hybrid domain with $\Omega_C = [1.0, 31.0]$ and $\Omega_M = [31.0, 41.0]$. The whole experiment is run 10 times from $t = 0$ s to $t_{\text{End}} = 30$ s with $\Delta t = 0.01$ s for better statistics.

Two illustrative values for k_{offset} will be simulated, namely $k_{\text{offset}} = \{8.5, 2.5\}$. Note that figure 6 of section 4.2 corresponds to setting the value $k_{\text{offset}} = \infty$, which means no voxel offset is present in the system.

The value $k_{\text{offset}} = 8.5$ is depicted in figure 8. This is the value obtained for c_2 found in section 3.3. There, the molecules mix-up well, and the effect of the interface I is diminished.

Lastly, in figure 9 one can see the interface at $I = 31.0$ for the distribution plot, and the effect that particles tend to move to the right to the particle domain Ω_M.

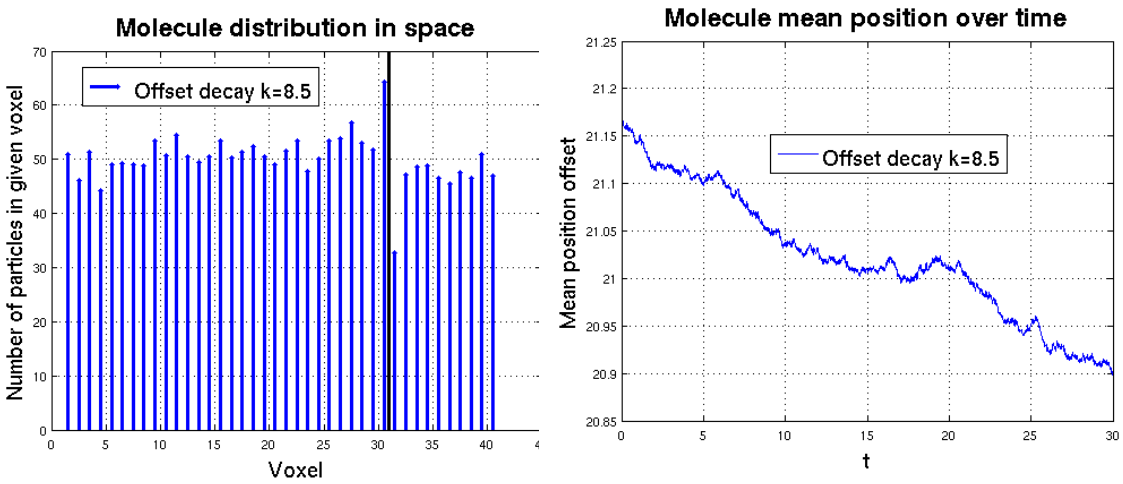

Figure 8: Distribution and mean offset for $k_{\text{offset}} = 8.5$

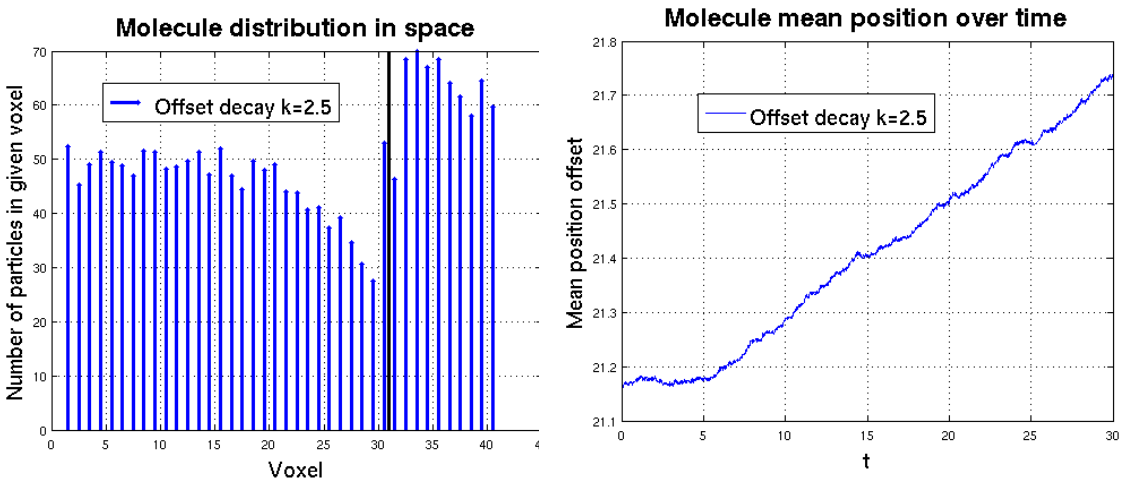

Figure 9: Distribution and mean offset for $k_{\text{offset}} = 2.5$

The particle mean position over time for the three different values is depicted in figure 10 for better comparison.

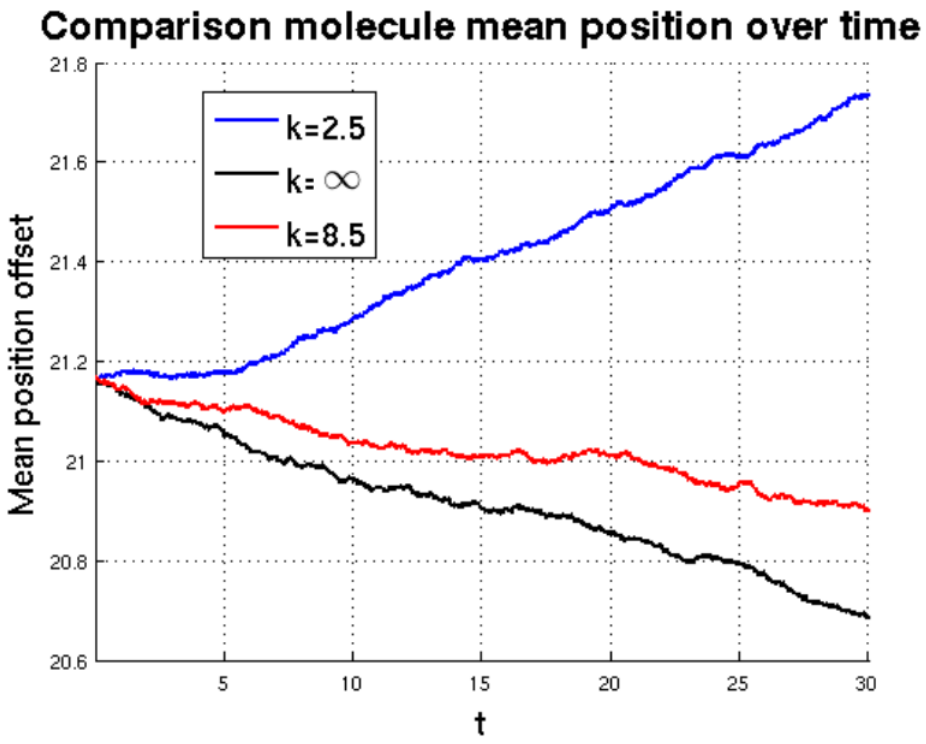

Figure 10: Comparison of the molecules mean position

5.3 Implementation and Results

The diffusion of $\mathbf{N}_{\text{particles}} = 10000$ particles for $t_{\text{End}} = 10$ s was simulated for 10 runs over a voxel domain of $\Omega_C = [1.0, 18.0]$ and a a particle domain of $\Omega_M = [18.0, 41.0]$.

Then, a comparison to the expected normalized distribution (6) was performed. The result can be seen on figure 11.

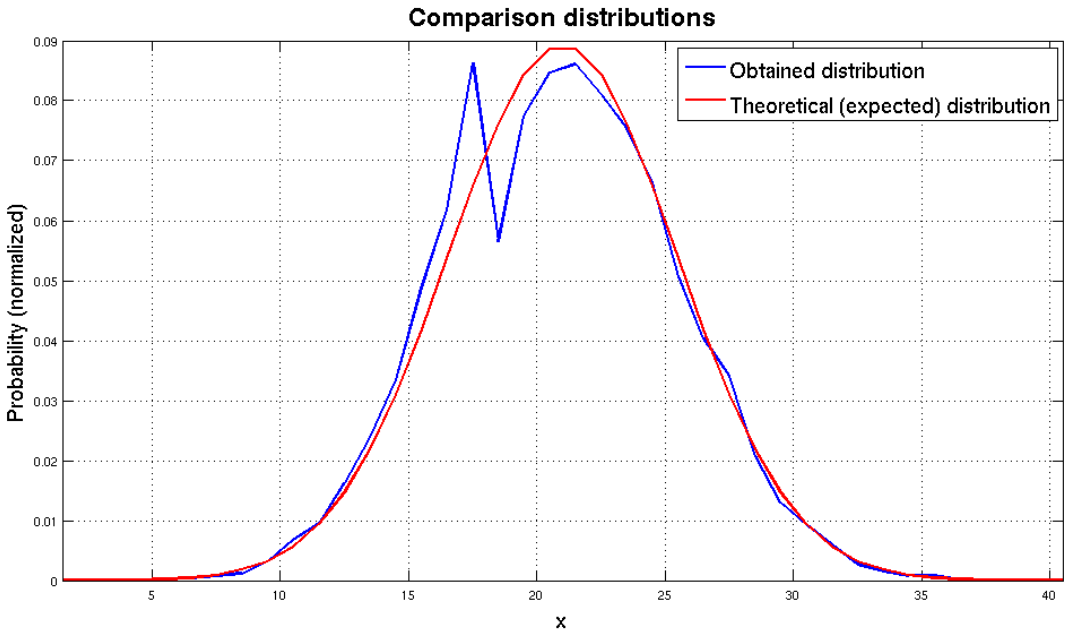

Figure 11: Theoretical normalized distribution (6) compared

The match to the theoretical distribution has no problems in terms of standard deviation and mean. However, near the interface $I = 18.0$ there is a clear undesired dent that differs from the result obtained in figure 3.

5.4 Remaining Problems, discussion, and comparison to the hybrid method

Although being beneficial for better interface treatment between regimes, the virtual adaptive voxel size still presents some drawbacks.

First, the simulation time t_{runtime} is increased with respect to the one of the original hybrid method.

Also, the dent observed in figure 11, which is related to the interface numbers of the interface adjacent voxels in the distribution on figure 8, is undesired. This problem has to be overcome, and a good starting point would be adjusting the time interval Δt and also t_{End}, or in other words, trying to terminate the simulation with lesser steps, thus avoiding those differences.

Another remaining task would be the refination and selection of the crucial decaying rate c_2. In this project it was assumed constant, and although working well for the settings used for the simulations. As can be seen in figure 10, the curve for $c_2 = 8.5$, although being better than the one for the hybrid model, is not perfect.

A method for choosing c_2, and hence the offset jumping rate constant k_{offset}, in a more precise manner regarding Δt and t_{End} is pending.

6 Conclusion and summary

The final improved method was built piece by piece by first implementing and simulating the basic previously existing models , namely the particle-based Brownian motion, the discrete space Gillespie method, and the hybrid method from Klann et al. [6].

The outcome of the experiments was satisfying to what was expected from theory. The particle distribution (6) was validated with a simulation, and despite remaining noise, mean and standard deviation were matching to the ones expected.

As was reasoned in section 3.3, there is a value for the decaying rate of a first order kinetics function, $c_2 = 8.5$, which matches the decaying of the mean offset for a single voxel in the particle-domain $\Omega_M = [0.0, 1.0]$. This value was used later on in the improved method.

In the hybrid method described on page 11, the obtained distribution after simulating a random standard uniform distribution presented a tendency for particles to move to the left. This motivated the improvement of the existing model [6] towards a correction of the artificial flux generated when dealing with an interface I, which was done in section §5.

In section §5, the theory and main principles of the virtual voxel length were introduced. This basically assures that for each voxel an average offset to the voxel center is stored. Also, the results of section 3.3 were re-used, since it is clear that the particle-domain $\Omega_M = [0.0, 1.0]$ corresponds to a voxel of size $h = 1.0$ in Gillespie-domain Ω_C. An improvement of the undesired flux could therefore be validated with the simulation shown in figure 8.

Unfortunately, figure 11 showed that the normalized distribution still has some issues. The particles tend to remain in Gillespie-domain voxels adjacent to the interface I, rather than jumping in the particle-domain near the interface. Again, the distribution is good, but the interface I seems to have a reflecting effect for particles jumping from Ω_C to Ω_M. This should be investigated and corrected in future projects.

Also, the found constant parameter c_2, while improving the mean offset, as can be seen in figure 10, still needs some tuning. Probably, c_2 has dependencies from Δt, and therefore the assumption of it being constant is too naive. System identification as well as optimization techniques should be used in future research to find a versatile method to compute the decaying value of the first order kinetics c_2.

Lastly, it would be very interesting to see how this improved method compares to the one used by Flegg et al. [3] by doing performance analysis and comparing the propensity of migration Φ and the probability Ψ to the virtual voxel length method presented here.

References

[1] Radek Erban, Jonathan Chapman, and Philip Maini. A practical guide to stochastic simulations of reaction-diffusion processes. November 2007.

[2] Mark B. Flegg, S. Jonathan Chapman, and Radek Erban. The two-regime method for optimizing stochastic reaction diffusion simulations. *Journal of The Royal Society Interface*, 9(70):859–868, October 2011.

[3] Mark B. Flegg, S. Jonathan Chapman, Likun Zheng, and Radek Erban. Analysis of the two-regime method on square meshes, April 2013.

[4] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403 – 434, 1976.

[5] Boris N. Kholodenko, John F. Hancock, and Walter Kolch. Signalling ballet in space and time. *Nat Rev Mol Cell Biol*, 11(6):414–426, June 2010.

[6] Michael Klann, Arnab Ganguly, and Heinz Koeppl. Hybrid spatial gillespie and particle tracking simulation. *Bioinformatics*, 28(18):549–555, 2012.

[7] Y. Lazebnik. Can a biologist fix a radio? or, what i learned while studying apoptosis. *Biochemistry (Moscow)*, 69(12):1403–1406, December 2004.

[8] Jana Lipkov, Konstantinos C Zygalakis, S Jonathan Chapman, and Radek Erban. Analysis of brownian dynamics simulations of reversible bimolecular reactions. *SIAM Journal on Applied Mathematics*, 71(3):714–730, 2011.

List of Figures

List of Algorithms