Stochastic Simulation Algorithms & Benchmarks.

Vasily Ilin

May 3, 2019

Contents

1 Project Goal

It is common in biology to have chemical networks consisting of hundreds or thousands of chemical reactions, e.g. the B cell network [1]. It is often unfeasible to solve the resulting system of differential equations 3.2 deterministically. Hence, the need for fast stochastic simulation arises. The goal of the project is to write several algorithms for stochastically simulating chemical systems, based on the Gillespie's algorithm [3], extend them to the spatial setting [2], compare their asymptotic running time, and benchmark them against each other. The main concern of this work is algorithmic efficiency, both theoretical and observed.

2 Overview of the report

In section 3 I introduce chemical reactions, propensities, sampling, and the Master Equation. In section 4 I describe the four algorithms I implemented to stochastically simulate chemical reactions. Section 5 exhibits an output of the algorithms for two concrete chemical networks. In section 6 I discuss the results of benchmarking these algorithms against each other. In section 7 I introduce diffusion. In section 8 two spatial algorithms are described. Section 9 exhibits benchmarking results of spatial algorithms. Results of my work are discussed in section 10, and possibilities of further work are outlined in 10.3.

3 Stochastic Chemical Kinetics

3.1 Reactions

Biochemical reactions are the basic tool of describing processes in biology, such as protein production by DNA. Consider the following notation: $A \xrightarrow{k} B$. This is a short way of writing that each molecule of type A can turn into a molecule of type B with some probability per time depending on k. In general, we will be concerned with a set of molecule types $S_1, ..., S_N$ and reactions $R_1, ..., R_M$ of the form

$$R_\mu : \sum_{n=1}^{N} v_n^- S_n \xrightarrow{k} \sum_{n=1}^{N} v_n^+ S_n, \tag{3.1}$$

where v_n^- and v_n^+ are integer valued. If $v_n^- \neq 0$, then S_n is called a reactant, and if $v_n^+ \neq 0$, S_n is called a product of the reaction. The vector $\vec{X}(t) = (X_1(t), ..., X_N(t))$ is called the state vector, where $X_n(t)$ is equal to the number of molecules of type S_n at time t. The vector $v_\mu = (v_1^+ - v_1^-, ..., v_N^+ - v_N^-)$ is called the net stoichiometry vector of reaction μ. It shows by how much the state $\vec{X}$ changes if the reaction fires once. Reactions of the form above are called mass-action reactions. I decided to focus exclusively on such reactions in my exploration, because they are the fundamental building blocks of most cellular chemistry.

3.2 Propensities

Referring to the example 3.1 above, k is called the reaction rate and is equal to the probability per time a single specific combination of reactant molecules reacts. Mathematically, $\mathbb{P}(E) = k\Delta t + o(\Delta t)$, where E is the event {a single pre-specified combination of reactant molecules reacts within time interval Δt}, and $\frac{o(\Delta t)}{\Delta t} \xrightarrow{\Delta t \to 0} 0$. Note that even though multiple reactions might occur in time period Δt, the probability of multiple reactions occurring is $o(\Delta t)$, so it is safe to assume that Δt is so small that only one reaction occurs. Then $a(\vec{x})\Delta t = kh(\vec{x})\Delta t + o(\Delta t) = \mathbb{P}$(the reaction occurs in time interval $[t, t + \Delta t]$), where $h(\vec{x})$ is the number of distinct combinations of reactants. We sum probabilities of two distinct combinations of molecules reacting, because the events are mutually exclusive, since Δt is so small that only one reaction can occur. Mathematically, $\mathbb{P}$(reaction μ occurs in interval $[t, t + \Delta t]$ $= 1 - \prod_{i=1}^{h(\vec{x})}(1 - k\Delta t + o(\Delta t)) = 1 - (1 - k\Delta t + o(\Delta t))^{h(\vec{x})} = 1 - (1 - h(\vec{x})k\Delta t + o(\Delta t)) = h(\vec{x})k\Delta t + o(\Delta t))$. For the example $A \xrightarrow{k} B$, $h(x) = m$, where m is the number of molecule of type A for state x. For $2A \xrightarrow{k} B$, $h(x) = \frac{m^2}{2}$.

3.3 Master Equation

Let $P(\vec{x}, t) = \mathbb{P}(\vec{X}(t) = \vec{x})$, where $\vec{X}(t)$ is the state vector at time t. Then $P(\vec{x}, t+\Delta t) = \mathbb{P}$(no reaction occurs$|\vec{X}(t) = \vec{x}) + \sum_{\mu}^{M} \mathbb{P}$(reaction μ occurs$|\vec{X}(t) = \vec{x} - v_\mu) + \mathbb{P}$(more than one reaction occurs in interval $[t, t + \Delta t]$), where v_μ is the stoichiometry vector of reaction μ. But $\mathbb{P}$(more than one reaction occurs in interval $[t, t + \Delta t]) = o(\Delta t)$, $\frac{o(\Delta t)}{\Delta t} \xrightarrow{\Delta t \to 0} 0$. So

$$P(\vec{x}, t + \Delta t) = (1 - \sum_{\mu=0}^{M} \Delta t a_\mu(x))P(\vec{x}, t) + \sum_{\mu=0}^{M} \Delta t a_\mu(\vec{x} - v_\mu)P(\vec{x} - v_\mu, t) + o(\Delta t)$$

Rearranging, and letting $\Delta t \to 0$, we get

$$\frac{dP(\vec{x},t)}{dt} = \sum_{\mu=0}^{M} a_\mu(\vec{x} - v_\mu)P(\vec{x} - v_\mu, t) - a_\mu(\vec{x})P(\vec{x},t) \tag{3.2}$$

Equation 3.2 is called the Master Equation. Moments of $\vec{X}(t)$ can be obtained from it by solving a system of ODEs corresponding to 3.2 (1 ODE per possible state $\vec{x}$). I state the ODE corresponding to k$^{\text{th}}$ moment without deriving it, as it is not relevant to this work:

$$\frac{d}{dt}\mathbb{E}[X^k] = \sum_{\mu=0}^{M} \mathbb{E}\big[((X + v_\mu)a_\mu(X))^k\big] - \mathbb{E}\big[(Xa_\mu(X))^k\big], \text{ where } X = \vec{X}(t)$$

3.4 Sampling

Fix time t and state $x(t)$. Let h_μ be the number of distinct combinations of reactants of reaction μ as above. Let $P_0(\tau) = \mathbb{P}(\text{no reaction occurs in time interval } [t, t+\tau])$. Then $P(\mu,\tau)\Delta t := \mathbb{P}(\text{reaction } \mu \text{ occurs in the interval } [t+\tau, t+\tau+\Delta t], \text{ and no other reaction occurs before it}) = k_\mu h_\mu \Delta t P_0(\tau) = a_\mu \Delta t P_0(\tau)$. Let ϵ, K be s.t. $\epsilon K = \tau$.

$$\mathbb{P}(\text{no reaction occurs in interval } [t + n\epsilon, t + (n+1)\epsilon]) = \prod_{\mu=1}^{M}(1 - a_\mu\epsilon + o(\epsilon)) = 1 - \sum_{\mu=1}^{M} a_\mu\epsilon + o(\epsilon),$$

where $\frac{o(\epsilon)}{\epsilon} \xrightarrow[\epsilon\to 0]{} 0$. Then

$$P_0(\tau) = \Big[1 - \sum_{\mu=1}^{M} a_\mu \frac{\tau}{K} + o((\frac{\tau}{K}))\Big]^K = \Big[1 - \frac{1}{K}\sum_{\mu=1}^{M} a_\mu\tau + o((\frac{\tau}{K}))\Big]^K \xrightarrow[K\to\infty]{} \exp\Big[-\sum_{\mu=1}^{M} a_\mu\tau\Big] \tag{3.3}$$

So $P(\mu,\tau) = a_\mu \exp(-a_0\tau) = \frac{a_\mu}{a_0} f_Y(\tau)$, where $a_0 = \sum_{\mu=1}^{M} a_\mu$, and $f_Y(\tau)$ is the probability density function of an exponentially distributed random variable Y, with mean $\frac{1}{a_0}$. Thus, in order to sample the time of reaction μ firing, in the absence of other reactions (that might change the propensity a_μ), need to sample the exponentially distributed random variable $Y \sim \text{Exp}(a_0)$.

4 Spatially Homogeneous Stochastic Simulation Algorithms (SSAs)

Often the Master equation cannot be solved analytically, because there is an ODE for each possible state of the system, resulting in either a very large system of ODEs or an infinite one. Stochastic simulation avoids this problem by computing exact trajectories of the system many times if necessary. Let $X_n^{(k)}(t) = \sum_{m=0}^{\infty} m^k \mathbb{P}(X_n(t) = m) = \mathbb{E}[X_n^k(t)]$ be the k$^{\text{th}}$ moment of the random variable $X_n(t) =$ the number of molecules of type S_n at time t. It can be shown that under normal conditions

$$\frac{1}{W}\sum_{w=1}^{W}(X_{n,w}(t))^k \xrightarrow[M\to\infty]{} X_n^{(k)}(t), \tag{4.1}$$

where $X_{n,w}(t)$ is the number of molecules of type S_n at time t in w^{th} simulation. In particular, taking the average of W simulations for large W gives a good approximation for the expectation of the number of molecules of type S_n at time t. If a stochastic method satisfies (4.1), it is called exact. All further algorithms are exact. The proofs of exactness are omitted but can be found in [4]

There are certain conditions that the chemical system needs to meet in order for the calculations above to hold. First, the system needs to stay well-mixed, i.e. the spatial distribution of molecules is uniform. Second, the reactions need to be instantaneous. The firs condition can be achieved if perfectly elastic collisions (molecules bouncing off each other) happen much more frequently than perfectly than perfectly inelastic collisions (molecules sticking together and reacting). I assume these conditions throughout this section.

4.1 Data Structures

There are a few data structures that are utilized by the algorithms to be described. First, there is the dependency graph.

Definition 4.1 (Dependency Graph) A Dependency Graph is a directed graph with the vertices $\{1, ..., M | M =$ number of reactions$\}$, and edges $\{(i,j) | \exists S_n \in \mathit{DependentSpecies}(R_i) \cap \mathit{Reactants}(R_j)\}$, where $\mathit{DependentSpecies}(R_i) = \{S_n | v_n^- \neq v_n^+\}$ is the set of molecule species that depend on reaction R_i, i.e. species whose number is changed when reaction R_i fires, and $\mathit{Reactants}(R_j) = \{S_n | v_n^- \neq 0\}$ is the set of reactants of reaction R_j. v_n^- and v_n^+ refer to equation (3.1).

Definition 4.2 (Binary MinHeap) A Binary MinHeap is a rooted tree, with the vertices $\{(1, \tau_1), ..., (M, \tau_M) | M =$ number of reactions$\}$. For vertex (μ, τ_μ), reaction index μ is the key, and τ_μ is the value. The value in each parent node has to be smaller or equal than the value in its children nodes. Each node has 2 children nodes and 1 parent node, unless one of the children nodes is a leaf node, in which case there might be only 1 child, i.e. it is a complete tree.

The most important features of Binary MinHeaps is that getting the minimum element is constant time, because it is the root, and inserting is $O(\log_2 M)$, where M is the number of items in the Heap.

4.2 Direct Method (Direct)

Direct Method is the simplest algorithm to be discussed here. I implemented it, as well as other non-spatial algorithms, using *Simulation Algorithms for Computational Systems Biology* [4] for the pseudocode. The idea of the method is to split each simulation step into two substeps. This is achieved via the observation that $\mathbb{P}($reaction j occurs in the interval $[t+\tau, t+\tau+\Delta t]) =$
$\mathbb{P}($some reaction occurs in the interval $[t+\tau, t+\tau+\Delta t]) \times \mathbb{P}($reaction j occurs $|$ some reaction occurs$)$. Starting at time t, the first substep is to draw the time τ, when the next reaction will happen, from $\text{Exp}(a_0(\vec{X}(0)))$, where a_0 is as in (3.3), i.e. a_0 is the sum of the propensities of all reactions . This is done via inverse sampling. The new time is then $t+\tau$. The second substep is to draw the reaction that will fire at time $t+\tau$. This is done via linear search over the reaction propensities $a_1, a_2, ..., a_M$, where $a_i = a_i(\vec{x})$ is the propensity of the i^{th} reaction, as a function of the current state $\vec{x}$. When reaction μ is chosen to fire, the current state is updated to be $\vec{x} + \vec{v_\mu}$, where $\vec{v_\mu}$ is the stoichiometry vector for reaction μ. The propensities are updated accordingly, and the process is repeated. The vanilla implementation of the Direct Method, all propensities are updated. However, in my implementation, the dependency graph is utilized, and only the dependent reactions are updated.

The Direct Method is powerful in its simplicity. It uses no complex data structures, and is easy to implement. It performs well in reaction networks with relatively small number of reactions. The *theoretical/asymptotic* bottleneck of the Direct Method is the substep of choosing the reaction to fire at the drawn time $t+\tau$. Since the search is linear, the overall complexity of the Direct Method is $O(M)$, where M is the number of reactions.

4.3 Next Reaction Method (NRM)

The Next Reaction Method is the second simplest algorithm, with asymptotic efficiency exponentially better than that of the Direct Method. The idea of the NRM is to maintain a priority queue of the tentative reaction times for all reactions. This is done via a Binary MinHeap, where each node has the reaction index as the key and the tentative reaction time as the value. Before the simulation begins, tentative firing times τ_μ are sampled from $\text{Exp}(a_\mu(\vec{X}(0)))$ for $\mu = 1, ..., M$, and inserted in a Binary MinHeap. Starting at time t, the smallest tentative reaction time τ, corresponding to reaction μ, is drawn from the heap. Reaction μ is executed in the same way as in the Direct Method: the current state is updated to be $\vec{x} + \vec{v_\mu}$. Instead of updating all propensities, NRM only updates propensities of reactions which share reactant species with the species that changed as a result of reaction μ firing. In other words, propensity a_j is recomputed after reaction μ fired if and only if the edge (μ, a_j) is in the Dependency Graph. Then the tentative times of the dependent reactions are sampled from $\text{Exp}(a_j)$ and inserted in the priority queue.

The Next Reaction Method utilizes more complex data structures, which need to be precomputed before the simulation and maintained throughout the simulation. The trade off is that NRM solves the problem of the inefficient linear search over the reaction propensities in order to choose the next reaction to fire. Instead, the bottleneck is now maintaining the heap, which is $O(D \log M)$, where D is the average number of dependent reactions in the dependency graph. Thus, NRM is $O(D \log M)$ asymptotically. However, it does not beat the DM for small or moderately sized reaction network because of the need to precompute and maintain complex data structures.

4.4 Rejection SSA (RSSA)

Rejection based SSA does not improve the asymptotic time of the simulation but focuses on eliminating the real biggest cost for Direct and NRM: updating the propensities after the state change. First, fix $c = 1.1$, or any other constant > 1 but close to 1. At the start of the simulation, the species bounds $(\underline{X}_n, \bar{X}_n) = (\lceil \frac{1}{c} X_n \rceil, \lceil cX_n \rceil)$ is computed for each species $S_n,\ n = 1, ..., N$. Let $\underline{x} = (\underline{X}_1, ..., \underline{X}_N)$, and $\bar{x} = (\bar{X}_1, ..., \bar{X}_N)$. The interval $(\underline{a}_\mu, \bar{a}_\mu)$ is computed, where $\underline{a}_\mu = a_\mu(\underline{x})$ and $\bar{a}_\mu = a_\mu(\bar{x})$. Since I am dealing with mass-action reactions only, all propensities are strictly increasing in the number of molecules X_n, so $\underline{a}_\mu = \min_{x \in (\underline{x}, \bar{x})} a_\mu(x),\ \bar{a}_\mu(x) = \max_{x \in (\underline{x}, \bar{x})} a_\mu(x)$. Plainly speaking, if each of the molecule species is within its respective interval $(\underline{X}_n, \bar{X}_n)$, the propensity of each reaction is within its respective interval $(\underline{a}_\mu, \bar{a}_\mu)$.

Let $\bar{a}_0(\vec{X}(t)) = \sum_{\mu=1}^{M} \bar{a}_\mu(\vec{X}(t))$. It will play the role of a_0 in the Direct method. At each step, time τ is sampled from $\text{Exp}(\bar{a}_0)$. Then, just like in Direct Method, the reaction to fire is chosen using linear search on the upper propensity bounds $\bar{a}_\mu$. Suppose reaction μ was chosen. Let r be a random number uniformly distributed in $[0, 1)$. If $\bar{a}_\mu \times r <= \underline{a}_\mu$, reaction μ fires. If $\underline{X}_n$, and $\bar{X}_n$ are close, then $\bar{a}_\mu$, and $\underline{a}_\mu$ are close, because propensities are continuous by (3.1). Else, the real propensity $a_\mu(x)$ is calculated, and if $\bar{a}_\mu \times r \leq a_\mu$, reaction μ fires, and time τ is incremented. If $\bar{a}_\mu \times r > a_\mu$, reaction μ is rejected (hence the name of the algorithm), the time τ is incremented to keep the algorithm exact, new time τ is sampled, and a new reaction is chosen via linear search. After reaction μ fires, the state is updated, and the check $\underline{X}_n \leq X_n \leq \bar{X}_n$ is performed for all n, s.t. molecule S_n is either a product or a reactant of reaction μ. Often, X_n is in bounds, and no propensity update is needed (and the speed up is gained). If X_n is out of bounds, the new species bounds $(\underline{X}_n, \bar{X}_n) = (\frac{1}{c} X_n, cX_n)$, and new propensity bounds $(\underline{a}_\mu, \bar{a}_\mu)$ are computed.

Using 3.1, and the fact that biological reactions almost always have 1 or 2 reactants, one can estimate the probability of immediate acceptance without having to compute the explicit propensity $P_{acc} = \underline{a}_\mu / \bar{a}_\mu = \underline{a}_\mu / a_\mu \times a_\mu / \bar{a}_\mu \approx c^{-2R_\mu}$ for X_n big enough that $\lceil cX_n \rceil \approx cX_n$, where $R_\mu =$ number of reactant species for reaction μ. Thus, $P_{acc} \approx 1 - c^{-2R_\mu} \approx 0.83$ or 0.68 for $R_\mu = 1$ and 2 respectively, and $c = 1.1$. Thus, the rejection step is not very costly, since in most cases the explicit propensity is not calculated.

Despite having the same worst case asymptotic complexity as the Direct Method, namely, $O(M)$, RSSA turned out to be the most efficient method I implemented, because it cuts down on the frequency of recomputing propensities, which is the most costly operation, according to my empirical findings, probably because of non-uniform memory access.

4.5 Composition-Rejection SSA (DirectCR)

While RSSA can be thought of as an attempt to improve the actual running time of the simulation, DirectCR brings the asymptotic time down. It utilizes the most complex data structures out of the four algorithms discussed. At the beginning of the simulation the composition step is performed: reaction μ is placed into group (not in the algebra sense) G_l if $2^{l-1} < a_\mu \leq 2^l$. Let $a^l = \sum_{\mu \in G_l} a_\mu$ be the propensity of group G_l. Thus, $\{G_l\}_{l \in \mathbb{Z}}$ is a partition of the set of reaction indices $\{1, ..., M\}$. First, time τ is sampled from $Exp(a_0)$, just as in the Direct Method. Then the group, where the next reaction will occur is sampled using linear search on a^l for all l. Suppose group l was chosen for some l. Let r_1, r_2 be random numbers uniformly distributed between 0 and 1, and N_l be the number of reactions in group G_l. Then $\mu = \lceil r_1 N_l \rceil$ is the potential reaction to fire. The rejection step is performed on μ: if $a_\mu > r_1 2^l$, reaction μ is accepted to fire. Else, new r_1, r_2 are drawn, and a new $a_\mu = \lceil r_1 N_l \rceil$ is tested against $r_1 2^l$. Note that $2^{l-1} < a_\mu \leq 2^l$ by construction, meaning the average number of rejections is bounded above by 2, because $\mathbb{P}(\text{reaction } \mu \text{ is accepted}) = \frac{a_\mu}{2^l} > \frac{1}{2}$. The state and propensities are updated accordingly. For each reaction λ that depends on reaction μ (using the dependency graph), the check on a_λ is performed to make sure no propensity is in the wrong group. Suppose reaction $\lambda \in G_{l'}$. If $2^{l'-1} < a_\lambda \leq 2^{l'}$ is false, reaction λ is placed in group G_l, for l such that $2^{l-1} < a_\mu \leq 2^l$. In other words, the group structure is maintained.

It is reasonable to assume that the number of groups needed throughout the simulation is bounded by $D << 100$. Then the cost of each jump of the simulation is $O(D) = O(1)$, because linear search on a^l is performed in order to select the group, where the next reaction fires. The rejection step is $O(1)$, because $\mathbb{P}(\text{reaction is accepted}) > \frac{1}{2}$, as shown above. Thus, DirectCR is asymptotically constant time. However, in my benchmarks, it is the slowest algorithm of the four, because swapping reactions between groups is costly, and the step of computing reaction propensities after each jump is still there.

5 Biological Results

Since the algorithms considered here are exact, it makes no sense to compare them in terms of their output. I used the NRM to stochastically simulate two biological reaction systems: the DNA repressor model and the multistate model. Figure 1 displays the DNA reaction network. The system of deterministic ODEs corresponding to the DNA model can be solved both analytically and numerically, but for bigger systems it is impossible. Figures 2 and 3 exhibit the exact trajectories of one simulation of each of the two models.

Figure 1: DNA network

$$\begin{aligned}
DNA &\xrightarrow{k_1} mRNA + DNA \\
mRNA &\xrightarrow{k_2} mRNA + P \\
mRNA &\xrightarrow{k_3} 0 \\
P &\xrightarrow{k_4} 0 \\
DNA + P &\xrightarrow{k_5} DNAR \\
DNAR &\xrightarrow{k_6} DNA + P
\end{aligned}$$

Figure 2: DNA repressor model

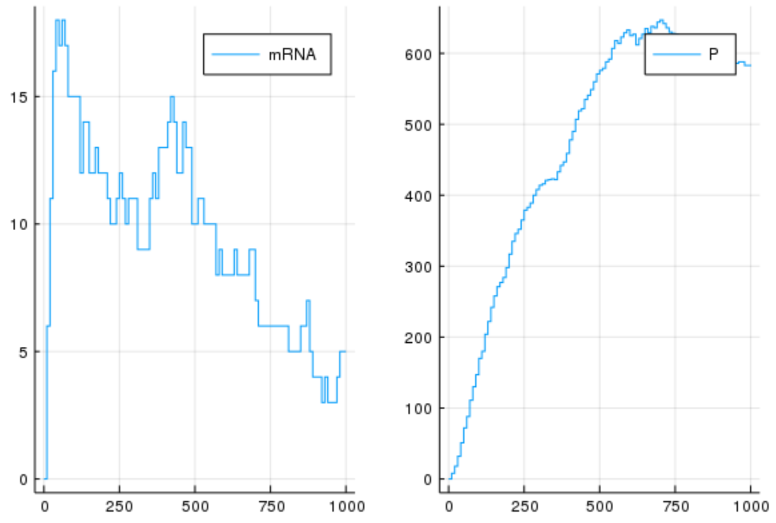
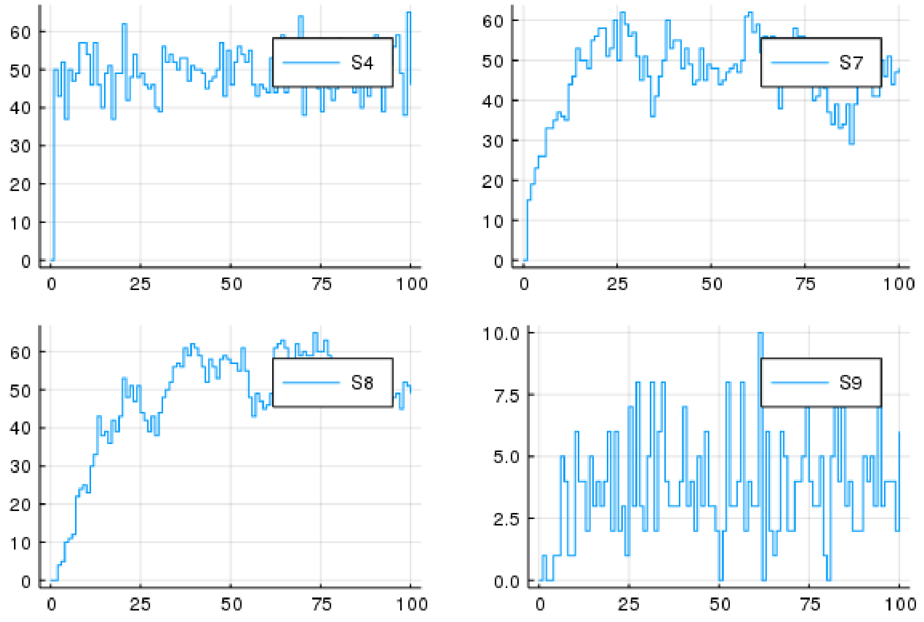


Figure 3: Multistate model



The trajectories give a sense of the behavior of the system. For example, it is clear that the steady state of the number of protein molecules in the DNA repressor model is around 600. By running multiple simulations one can calculate statistics about the system, such as mean, variance, etc.

6 Benchmarks

Section 4 focused on asymptotic performance. Based on big-O analysis, asymptotically, the highest costs of each algorithm are as follows:

Table 1: Theoretical highest costs

Direct	NRM	RSSA	DirectCR
reaction sampling , $O(M)$	reaction sampling, $O(\log M)$	reaction sampling, $O(M)$	group sampling, $O(D) = O(1)$

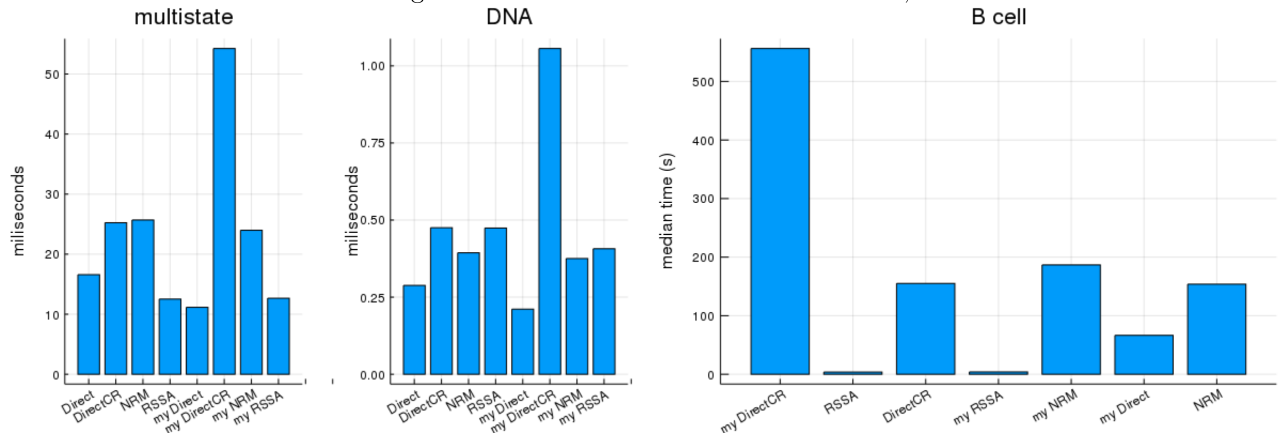
This section focuses on the actual run time of the four algorithms on three biological networks of different size.
I benchmarked the performance of each of the four methods, across three models. The models have 4, 9, and 1122 molecule species respectively. The numbers of reactions are 6, 18, and 24388 respectively. The biology of the models is not pertinent to the discussion, so I will simply refer to them as “DNA”, “multistate”, and “B cell,” ordered by size in ascending order. The median times for one simulation are summarized in the tables below. See the Appendix for more detailed information on the number of runs, minimum, maximum and mean times.

Table 2: Median simulation times

Method / Model	Direct	NRM	RSSA	DirectCR
DNA	$166\mu s$	$291\mu s$	$333\mu s$	$847\mu s$
multistate	$10ms$	$23ms$	$12ms$	$52ms$
B cell	$65s$	$185s$	$4s$	$556s$

The data from table 2 is visualized in figure 4, with comparison against the native DiffEqJump’s algorithms.

Figure 4: Median times for the three models,



Using the @profile command in Julia, which tracks how much time is spent on each individual line of the code, I found out the most costly operation for each of the four algorithms.

Table 3: Empirical highest costs

Direct	NRM	RSSA	DirectCR
updating propensities	updating propensities	sampling firing time and reaction	swapping reactions between groups

Note that RSSA is the only algorithm whose theoretical bottleneck is also its empirical bottleneck. There might be a better way to implement DirectCR, where swapping reactions between groups is less costly (I use dictionaries, which are poorly optimized in Julia, as opposed to arrays). So more testing would be needed to find the best implementation of DirectCR, and see what the limiting operation is.

7 Spatially Heterogeneous Stochastic Chemical Kinetics

The preceding methods aim to simulate a well-mixed spatially homogenous system. Clearly, it is simplification of reality, where diffusion and clustering of molecules at one region may occur.

For simplicity, consider a 1D system with one molecule species X and no reactions. Suppose molecule X can either hop h length units left or right, or only left or right if it is located at a boundary site. Then each hop can be viewed as a reaction, and the previous theory applies. Let X_i denote the number of molecules at site i, and let J be the number of sites. Then we have the following diagram:

$$X_0 \overset{\beta}{\leftrightarrow} X_1 \overset{\beta}{\leftrightarrow} ... \overset{\beta}{\leftrightarrow} X_{J-1} \overset{\beta}{\leftrightarrow} X_J \tag{7.1}$$

In other words, each spatial hop $X_i \xrightarrow{\beta} X_{i+1}$ can be thought of as a reaction with one reactant molecule X_i and one reaction product X_{i+1}. Here, β is the diffusion rate, and the propensity of hop from site i to site $i \pm 1$ is given by βX_i.

It is shown in section 11.1 that for the continuous time Random Walk to converge to Brownian motion, we must have $\beta h^2 \xrightarrow{\beta\to\infty,\ h\to 0} D$, where $0 < D < \infty$. D is called the diffusion constant, and should be thought of as an invariant physical constant that can be measured empirically for each molecule species individually. I will assume that D is the same for all diffusing molecule species, and is given. Hence, I set the diffusion rate $\beta := \frac{D}{h^2}$ in my simulations.

For the 2D (3D) case, each interior site will have 4 (6) neighboring sites instead of two, but nothing else changes. I will always assume that the volume where reactions and diffusion happen is of size 1, i.e. unit interval, unit square, and unit cube for 1D, 2D, and 3D, respectively. Even though the total number of sites is $\frac{1}{h^3}$, I will refer to $J = \frac{1}{h}$ as the number of sites. For example, figure 5 shows what the 2D case looks like for $J = 3$.

Figure 5: 2D diffusion

$$\begin{tikzcd}
X_1 \arrow[r, leftrightarrow, "\beta"] \arrow[d, leftrightarrow, "\beta"'] & X_2 \arrow[r, leftrightarrow, "\beta"] \arrow[d, leftrightarrow, "\beta"'] & X_3 \arrow[d, leftrightarrow, "\beta"'] \\
X_4 \arrow[r, leftrightarrow, "\beta"] \arrow[d, leftrightarrow, "\beta"'] & X_5 \arrow[r, leftrightarrow, "\beta"] \arrow[d, leftrightarrow, "\beta"'] & X_6 \arrow[d, leftrightarrow, "\beta"'] \\
X_7 \arrow[r, leftrightarrow, "\beta"'] & X_8 \arrow[r, leftrightarrow, "\beta"'] & X_9
\end{tikzcd}$$

8 Spatial Stochastic Simulation Algorithms

There are three conditions that need to be satisfied for spatial stochastic simulation, the way I implemented it, to make sense. First, we no longer assume that the whole volume is well mixed but each subvolume still has to be well-mixed. Second, reactions are instantaneous. And third, only molecules within one subvolume can react. Physically, this is achieved by letting the mesh size h be much larger than reaction distance.

I will still consider mass-action reactions of the form (3.1), but now there are J state vectors, one for each subvolume J.

8.1 Data Structures

The spatial algorithms to be discussed utilize two data structures: a connectivity matrix and a rate matrix.

The connectivity matrix is a $J^d \times 2d$ matrix, where $d = 1, 2, 3$ is the dimension, and J is the number of sites. Each row vector v_j contains the neighbors of site j. Figure 6 shows an example of a connectivity matrix for the system in figure 5. Note that some entries are $*$, because only interior sites have $2d$ neighbors.

The rate matrix is a $J^d \times (M+N)$ matrix, where N, M are the number of species and number of reactions, respectively. Row vector j stores reaction and diffusion propensities for site j. If $a_{rx}^{j,\mu}$ = propensity of reaction μ at site j, and $a_{diff}^{j,n}$ = propensity of species n at site j hopping to a neighboring site, then row vector j will be given by $(a_{rx}^{j,1}, ..., a_{rx}^{j,M}, a_{diff}^{j,1}, ..., a_{diff}^{j,N})$. I will call $a_{rx}^{j} = a_{rx}^{j,1} + ... + a_{rx}^{j,M}$ the

Figure 6: connectivity matrix example

$$\begin{bmatrix} 2 & 4 & * & * \\ 1 & 3 & 5 & * \\ 2 & 6 & * & * \\ 1 & 5 & 7 & * \\ 2 & 4 & 6 & 8 \\ 3 & 5 & 9 & * \\ 4 & 8 & * & * \\ 5 & 7 & 9 & * \\ 6 & 8 & * & * \end{bmatrix}$$

total reaction propensity of site j, and $a^j_{diff} = a^{j,1}_{diff} + ... + a^{j,N}_{diff}$ the total diffusion propensity of site j. The total propensity of site j is given by $a_j = a^j_{rx} + a^j_{diff}$.

8.2 Spatial Direct and Spatial NRM

For simplicity, consider the 1D case first. Suppose the molecules are all in an interval $[0,1] \subset \mathbb{R}$. Since the distribution of molecules is no longer assumed to be uniform, the probability of a reaction happening in, say, $[0, 0.5]$ is not equal to the probability of a reaction happening in $[0.5, 1]$. Thus, we partition $[0,1]$ into intervals $[nh, (n+1)h]$ for $n = 1, ..., J,\ J \times h = 1$, and keep track of the state of the system in each interval separately. For each chemical system there is the set of molecule species that is assumed to be capable of diffusion, and the set of molecule species that is assumed to be fixed. This is a simplification of the model, where each species is given its own diffusion constant, which is a function of the physical properties of the molecule.

The simplest model I am considering is the DNA repressor network. It involves four species: DNA, repressed DNA, mRNA, and protein. DNA and repressed DNA molecules are assumed to be fixed, while mRNA and protein molecules are free to diffuse. I will describe how the spatial versions of the Direct Method and the NRM differ from their non spatial analogues.

The Spatial Direct Method has three steps in the main simulation loop. First, time τ is drawn from $\text{Exp}(a_1 + ... + a_{J^d})$, where d is the dimension. Then DM linearly chooses the site to react, using the total propensities of each individual site. Second, DM decides if a reaction or a diffusion occurs at the chosen site j by comparing $r \times a_j$ with a^j_{rx}, where r is a random number between 0 and 1 (drawn from uniform distribution). If $r \times a_j < a_{rx}$, reaction occurs. Otherwise, diffusion occurs. Third, DM linearly chooses the reaction or species index to react or diffuse and executes the reaction or diffusion. The current state is updated accordingly, and propensities are recomputed, using the dependency graph. Spatial DM is $O((N + M)J^{\text{dimension}})$, where N is the number of species, M is the number of reactions, and J is the number of subvolumes. It is linear in N (= # of molecule species) because if a diffusion occurs, the species to diffuse is chosen linearly from N species. It is linear M (= # of reactions) because if a reaction occurs, the reaction to fire is chosen linearly from M reactions. It is linear in $J^{\text{dimension}}$ (= total number of sites) because the choice of site is also linear.

The Spatial Next Reaction Method mimics the non spatial NRM described earlier. Instead of holding tentative reaction times in the priority queue, tentative subvolume activation times are held in the queue. This allows for logarithmic time in the number subvolumes. Once the site with the smallest tentative volume activation time is chosen, the method starts behaving just like the Spatial DM: the reaction to fire or species to diffuse is chosen linearly. Thus, Spatial NRM is $O((N + M) \log J)$. In particular, Spatial NRM is logarithmic in the number of subvolumes.

Since the Spatial DM is linear in the number of subvolumes, and the Spatial NRM is logarithmic in the number of subvolumes, it is reasonable to expect bigger and bigger discrepancy between the running times of the two methods, as the number of subvolumes increases. Indeed, such result was obtained.

9 Spatial Methods Benchmarks

I benchmarked Spatial Direct and Spatial NRM against each other using the DNA model and the multistate model, for $J = 11, 21, 31, 41, 51, 61$. The results are summarized in the following figures. Note that by "number of subvolumes" I mean J, and not $J^{\text{dimension}}$ The raw data can be found in the Appendix, section 11.2.

In order to explicitly recover the difference between the Spatial Direct and NRM methods, I fitted a linear polynomial to the DNA 3D run time data from table 4, row 3. Figure 9 shows that the fit is almost perfect. Let $T_d(J),\ T_n(J)$ be the average simulation time of the Direct and NRM, respectively as a function of the linear number of subvolumes J (with J^3 total sites). Then

$$\log(T_d(J)) \approx -14 + 5\log(J) \implies T_d(J) \approx e^{-14} \times J^5 = e^{-14} \approx (J^3)^{1.7}, \text{ where } J^3 \text{ is the total number of sites.} \tag{9.1}$$

$$\log(T_n(J)) \approx -12.5 + 3.8\log(J) \implies T_n(J) \approx e^{-12.5} \times J^{3.8} \approx e^{-12.5} \times (J^3)^{1.27}, \text{ where } J^3 \text{ is the total number of sites.} \tag{9.2}$$

We see that Spatial Direct is indeed almost linear in the total number of sites J^3, and from profiling, the bottleneck of Direct is the linear search, as predicted. However, Spatial NRM is not actually logarithmic in the total number of sites, because updating propensities becomes the limiting operation, because of non-uniform memory access.

Figure 7: DNA model

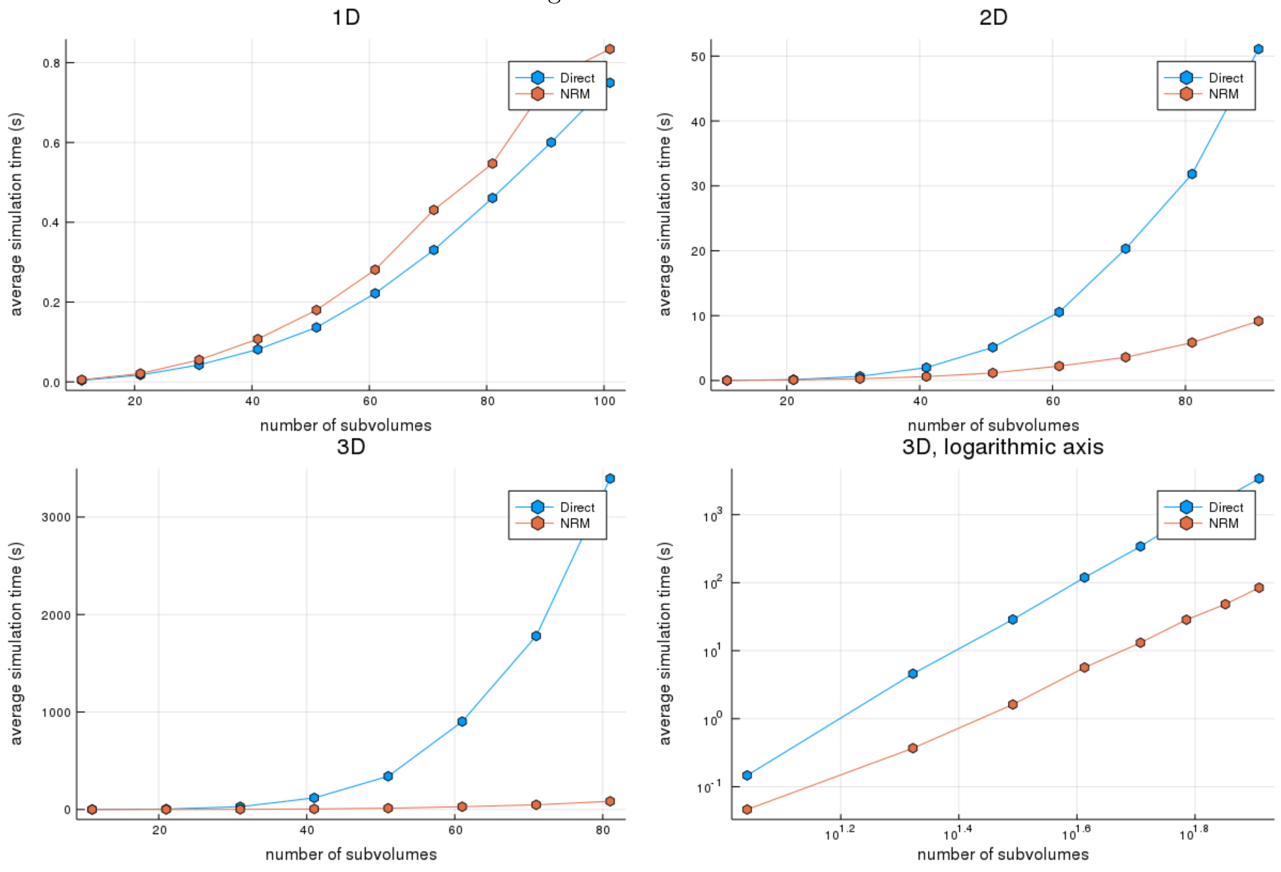
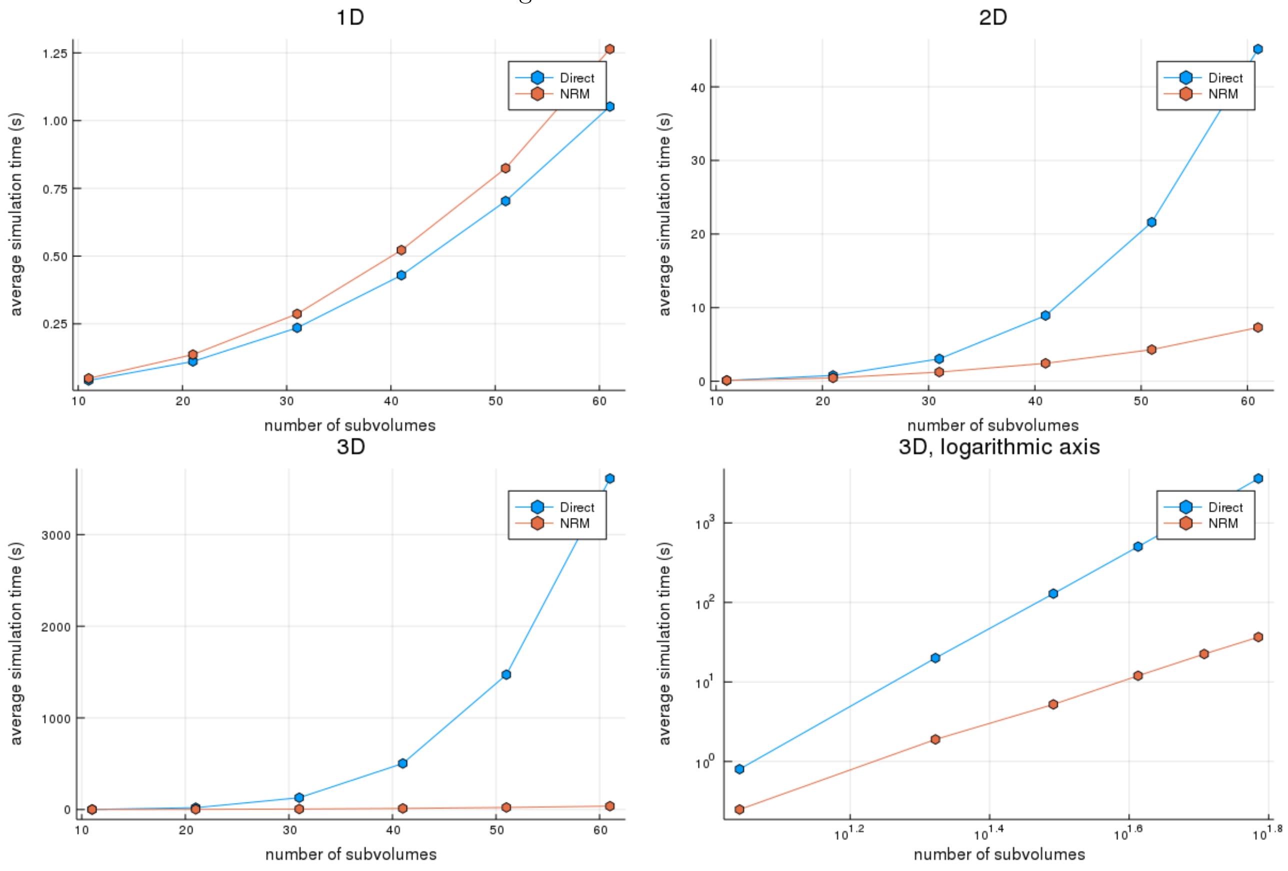


Figure 8: multistate model

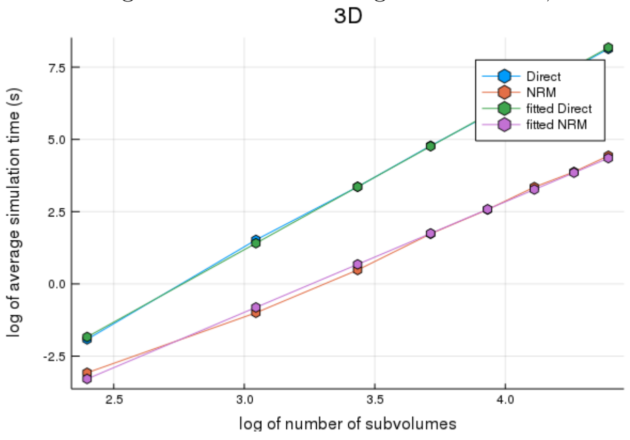


10 Discussion of Results

10.1 Non-Spatial SSA's

My findings showed that there is a big discrepancy between what big-O analysis predicts to be the bottleneck and the empirical data. Specifically, I found out that the algorithm with the best asymptotic time performed the worst of the three networks I used for benchmarking. This is surprising, and probably is partially due to my imperfect implementation of the algorithm. However, it is clear that in order to improve the efficiency of stochastic simulation algorithm, it is not enough to only consider asymptotic run time.

Figure 9: DNA fitted logarithmic data,



When analyzing the run time of stochastic simulation on medium sized networks, the limiting factor is often not sampling the reaction to fire and firing time but updating propensities. I hypothesize that the reason for it is the way memory is accessed. For example, consider the B cell network. There are ≈ 25000 reactions, so the propensity vector is of length 25000, and of size $\approx 190Kb$, meaning it fits in the cache memory. So when a linear search is performed, each individual access in very inexpensive. When updating the propensity of reaction μ, a call is made to `evalrxrate`, the function that evaluates one propensity. The average number of dependent reactions $D = 380$ for this network, meaning RAM is accessed over 380 times, in a non-uniform manner. I expect that for reaction networks with the number of reactions so high that the propensity vector does not fit in cache, the limiting operation for Direct method and RSSA will be reaction sampling, and NRM and DirectCR will stand out more. However, even with the network size of 25000 reactions, it takes 5 seconds of real time to simulate 1 second of simulated time using DirectCR. Thus, if there were hundreds of thousands or millions of reactions (so that the propensity vector does not fit in cache), simulation using any algorithm would become prohibitively slow.

10.2 Spatial SSA's

The results of benchmarking the spatial analogues of Direct and NRM show that Spatial Direct is indeed almost linear in the number of sites, with reaction sampling being the limiting operation. However, NRM is not actually logarithmic in the number of sites, as shown in (9.2). Profiling Spatial NRM showed that the limiting operation is not reaction sampling but propensity updating, similar to the non spatial analogues. However, the difference in the run time of Spatial Direct and Spatial NRM is drastic when the total number of sites $J^{\text{dimension}}$ is higher than $\approx 40^3$. This motivates further work on different ways to optimize spatial stochastic simulation using spatial analogues of RSSA and DirectCR.

10.3 Further Work

In spatial simulation, there are two basic steps for sampling the reaction to fire or molecule species to diffuse: sampling the site, and sampling the reaction or diffusion to occur at the site. Thus, one method can be used for drawing the site and another method – for drawing the reaction/species. For example, since DirectCR has constant asymptotic run time, DirectCR can be used to sample the site (each site will be placed in its respective group G_l, based on its propensity a_j). This will allow for very fine mesh, even in 3D. And since RSSA performs the best in the non-spatial setting, drawing the reaction/species can be done with RSSA.

Another possible direction of further work is allowing reactions to happen not only within one site but across neighboring site, or in a whole neighborhood $B_r(x)$, where $x \in \mathbb{R}^3$, and r is the physical reaction distance of approximately $5nm$ [2].

References

[1] Dipak Barua, William S Hlavacek, and Tomasz Lipniacki. "A computational model for early events in B cell antigen receptor signaling: analysis of the roles of Lyn and Fyn". In: *The Journal of Immunology* 189.2 (2012), pp. 646–658.

[2] Johan Elf and Måns Ehrenberg. "Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases". In: *Systems biology* 1.2 (2004), pp. 230–236.

[3] Daniel T Gillespie. "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions". In: *Journal of computational physics* 22.4 (1976), pp. 403–434.

[4] Luca Marchetti, Corrado Priami, and Vo Hong Thanh. *Simulation algorithms for computational systems biology.* Springer, 2017.

11 Appendix

11.1 Random Walk

I only consider the 1D case for simplicity. From 7.1 and 3.2, the Master equation for 1D diffusion of 1 molecule is

$$\frac{dP_i}{dt} = \beta(P_{i+1} - 2P_i + P_{i-1}), \text{ where } P_i(t) = \mathbb{P}(\text{molecule is at position } x_i = ih)$$

To get a probability density, rescale the ODE: $\rho(x_i, t) = \frac{P_i(t)}{h}$. Then

$$\frac{d\rho(x_i, t)}{dt} = \beta(\rho(x_{i+1}, t) - 2\rho(x_i, t) + \rho(x_{i-1}, t)) \tag{11.1}$$

But $\rho(x_{i\pm1}, t) = \rho(x_i \pm h, t) = \rho(x_i, t) \pm h\frac{\partial\rho}{\partial x}(x_i, t) + \frac{h^2}{2}\frac{\partial^2\rho}{\partial x^2}(x_i, t) \pm \frac{h^3}{6}\frac{\partial^3\rho}{\partial x^3}(x_i, t) + o(h^3)$. Hence, 11.1 becomes:

$$\frac{d\rho(x_i, t)}{dt} = \beta h^2 \frac{\partial^2\rho}{\partial x^2}(x_i, t) + o(h^3)$$

In order to get a nontrivial limit for $\frac{d\rho(x_i,t)}{dt}$, need $\beta h^2 \xrightarrow{\beta\to\infty,\ h\to 0} D$, where $0 < D < \infty$. Then

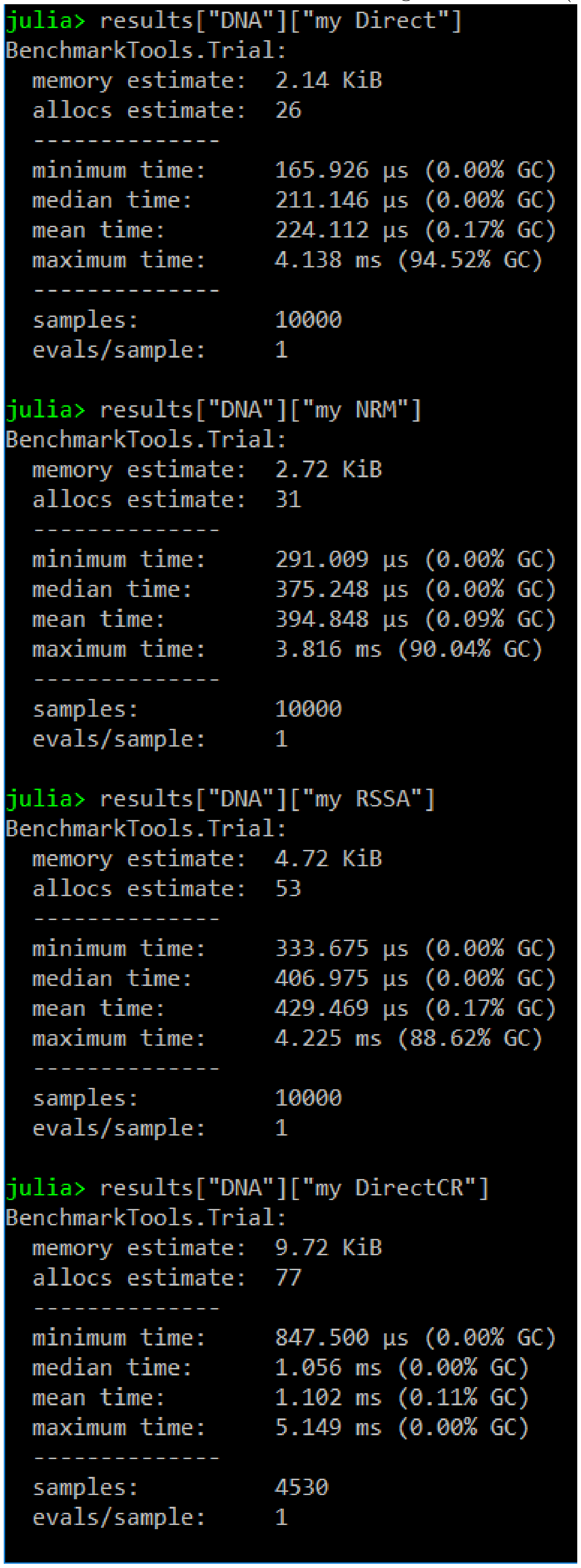
$$\frac{d\rho(x, t)}{dt} = \beta h^2 \frac{\partial^2\rho}{\partial x^2}(x, t) = D\frac{\partial^2\rho}{\partial x^2}(x, t)$$

In other words, if $\beta h^2 \xrightarrow{\beta\to\infty,\ h\to 0} D$, then the space discretization converges to the Brownian motion for decreasing mesh size, which is what we want.

11.2 Benchmarks Data

Benchmarks of my DM, NRM vs pre-built DM, NRM in Julia on the DNA repressor and the multistate models.

Figure 10: DNA (left), multistate (right)

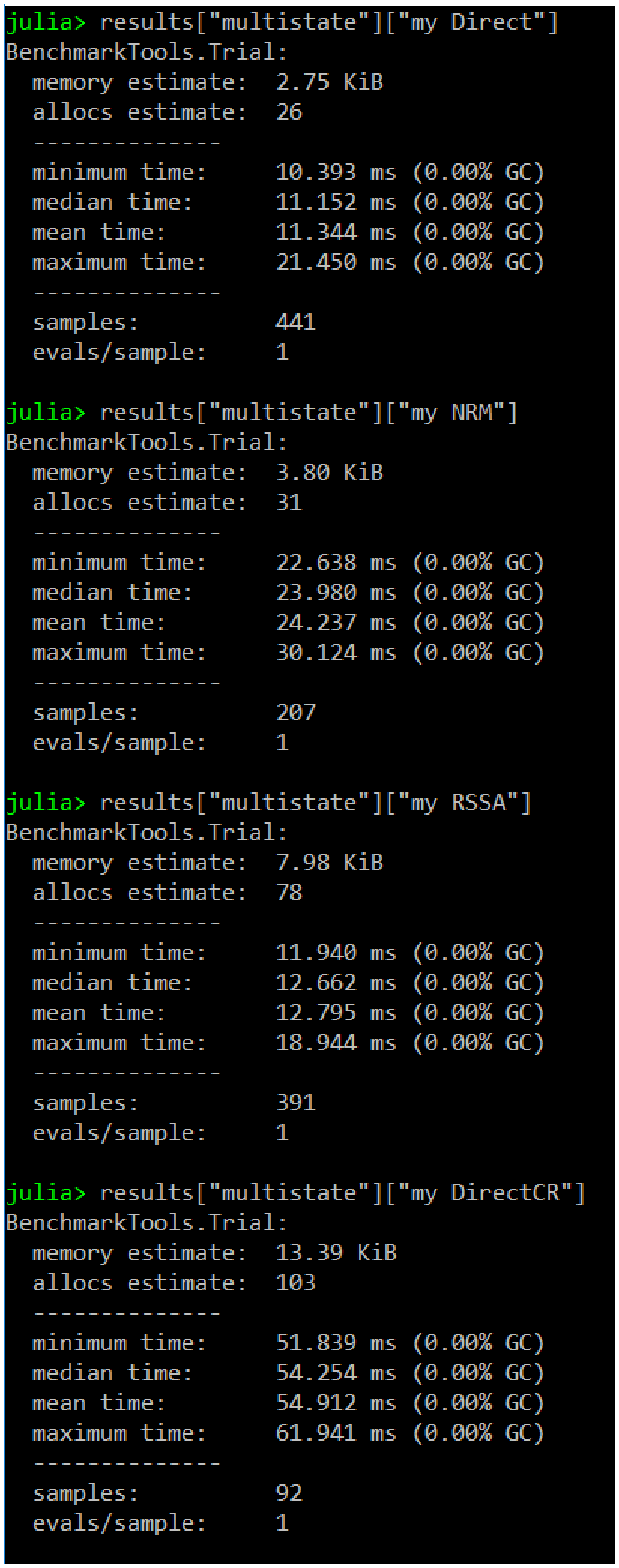


```
julia> results["DNA"]["my Direct"]
BenchmarkTools.Trial:
  memory estimate:  2.14 KiB
  allocs estimate:  26
  --------------
  minimum time:     165.926 μs (0.00% GC)
  median time:      211.146 μs (0.00% GC)
  mean time:        224.112 μs (0.17% GC)
  maximum time:     4.138 ms (94.52% GC)
  --------------
  samples:          10000
  evals/sample:     1

julia> results["DNA"]["my NRM"]
BenchmarkTools.Trial:
  memory estimate:  2.72 KiB
  allocs estimate:  31
  --------------
  minimum time:     291.009 μs (0.00% GC)
  median time:      375.248 μs (0.00% GC)
  mean time:        394.848 μs (0.09% GC)
  maximum time:     3.816 ms (90.04% GC)
  --------------
  samples:          10000
  evals/sample:     1

julia> results["DNA"]["my RSSA"]
BenchmarkTools.Trial:
  memory estimate:  4.72 KiB
  allocs estimate:  53
  --------------
  minimum time:     333.675 μs (0.00% GC)
  median time:      406.975 μs (0.00% GC)
  mean time:        429.469 μs (0.17% GC)
  maximum time:     4.225 ms (88.62% GC)
  --------------
  samples:          10000
  evals/sample:     1

julia> results["DNA"]["my DirectCR"]
BenchmarkTools.Trial:
  memory estimate:  9.72 KiB
  allocs estimate:  77
  --------------
  minimum time:     847.500 μs (0.00% GC)
  median time:      1.056 ms (0.00% GC)
  mean time:        1.102 ms (0.11% GC)
  maximum time:     5.149 ms (0.00% GC)
  --------------
  samples:          4530
  evals/sample:     1
```



```
julia> results["multistate"]["my Direct"]
BenchmarkTools.Trial:
  memory estimate:  2.75 KiB
  allocs estimate:  26
  --------------
  minimum time:     10.393 ms (0.00% GC)
  median time:      11.152 ms (0.00% GC)
  mean time:        11.344 ms (0.00% GC)
  maximum time:     21.450 ms (0.00% GC)
  --------------
  samples:          441
  evals/sample:     1

julia> results["multistate"]["my NRM"]
BenchmarkTools.Trial:
  memory estimate:  3.80 KiB
  allocs estimate:  31
  --------------
  minimum time:     22.638 ms (0.00% GC)
  median time:      23.980 ms (0.00% GC)
  mean time:        24.237 ms (0.00% GC)
  maximum time:     30.124 ms (0.00% GC)
  --------------
  samples:          207
  evals/sample:     1

julia> results["multistate"]["my RSSA"]
BenchmarkTools.Trial:
  memory estimate:  7.98 KiB
  allocs estimate:  78
  --------------
  minimum time:     11.940 ms (0.00% GC)
  median time:      12.662 ms (0.00% GC)
  mean time:        12.795 ms (0.00% GC)
  maximum time:     18.944 ms (0.00% GC)
  --------------
  samples:          391
  evals/sample:     1

julia> results["multistate"]["my DirectCR"]
BenchmarkTools.Trial:
  memory estimate:  13.39 KiB
  allocs estimate:  103
  --------------
  minimum time:     51.839 ms (0.00% GC)
  median time:      54.254 ms (0.00% GC)
  mean time:        54.912 ms (0.00% GC)
  maximum time:     61.941 ms (0.00% GC)
  --------------
  samples:          92
  evals/sample:     1
```

Benchmarks of Spatial methods on DNA and multistate models

Table 4: DNA Direct average times,

J:	11	21	31	41	51	61	71	81
1D	0.0036	0.0173	0.0426	0.0814	0.1364	0.2218	0.3306	0.4609
2D	0.0167	0.1436	0.6562	1.9893	5.1146	10.5466	20.3146	31.8296
3D	0.1467	4.5828	28.7611	119.0822	340.9756	901.8164	1779.8964	3395.0829

Table 5: DNA NRM average times

J:	11	21	31	41	51	61	71	81
1D	0.0053	0.0209	0.0551	0.1073	0.18	0.2811	0.4309	0.5472
2D	0.0166	0.0955	0.2765	0.6083	1.1678	2.2271	3.5876	5.8509
3D	0.0462	0.3683	1.6174	5.6555	13.0847	28.5119	48.1867	84.2018

Table 6: multistate Direct average times

J:	11	21	31	41	51	61
1D	0.0411	0.1111	0.2353	0.4292	0.703	1.0517
2D	0.11	0.7899	3.0403	8.9264	21.6046	45.1272
3D	0.8001	19.978	128.895	502.8747	1471.3703	3611.8385

Table 7: multistate NRM average times

J:	11	21	31	41	51	61
1D	0.0487	0.1365	0.2867	0.522	0.8242	1.2639
2D	0.1172	0.4523	1.2465	2.4358	4.2962	7.3023
3D	0.2489	1.8925	5.2188	11.9603	22.4443	36.6426