Rajesh Ramaswamy & Ivo F. Sbalzarini

# Chemical Reaction Network Simulation

Lecture Notes

**TU Dresden, Faculty of Computer Science**
**Chair of Scientific Computing for Systems Biology**

Prof. Dr. Ivo F. Sbalzarini, TUD & MPI-CBG
Dr. Rajesh Ramaswamy, MPI-PKS

October 2015

This page is intentionally left blank

# Contents

# 1 Introduction

*In this chapter:*

- What are reaction networks?
- Representation and types of reactions
- Network invariants and conservation laws
- Graphical representation of reaction networks
- Classification of reaction networks

*Learning goals:*

- Be able to define "reaction" and "reaction network"
- Be able to write down the stoichiometric matrices of reaction networks
- Be able to compute network invariants and conservation laws
- Be able to represent reaction networks by their dependency graphs
- Be able to decide the class of a network based on its dependency graph

**Definition 1:** A *reaction* is any process where an interaction between two or more entities leads to a transformation of at least one of them.
**Definition 2:** A *reaction network* is a set of reactions that share at least one species each. That is, for each reaction there exists at least one other reaction in the set with at least one species in common.

## 1.1 Representation of a reaction

In reaction notation: $A + B \to C$. In general *reactants* $\to$ *products*:

$$\sum_{i=1}^{N} \nu_i^- S_i \xrightarrow{k} \sum_{i=1}^{N} \nu_i^+ S_i \,, \tag{1.1}$$

where:

$S_i$: species $i$

$N$: total number of different species in the reaction

$\nu_i^-$: reactant stoichiometry

$\nu_i^+$: product stoichiometry

$k$: reaction rate

The *total stoichiometry* is $\nu_i = \nu_i^+ - \nu_i^-$, and it gives the net change in molecule numbers when the reaction happens. Reactions are classified by the total number of reactant molecules $\sum_i \nu_i^-$, which is called the *order* of the reaction. Reactions having only a single reactant are of order one, or *unimolecular*. Reactions with two reactants are of order two, or *bimolecular*; and so on.
Example: for the reaction $A + B \to C$, the above quantities are:

- $S_i = \{A, B, C\}$
- $N = 3$
- $\boldsymbol{\nu}^- = [1, 1, 0]^\top$
- $\boldsymbol{\nu}^+ = [0, 0, 1]^\top$
- $\boldsymbol{\nu} = [-1, -1, 1]^\top$

## 1.2 Representation of a reaction network

$$\sum_{i=1}^{N} \nu_{i,\mu}^- S_i \xrightarrow{k_\mu} \sum_{i=1}^{N} \nu_{i,\mu}^+ S_i \,, \qquad \mu = 1, \ldots, M \tag{1.2}$$

where:

$\mu$: index of reaction $R_\mu$

$M$: total number of different reactions

Now the stoichiometry is a matrix with one column per reaction: $\underline{\underline{\nu}} = \underline{\underline{\nu}}^+ - \underline{\underline{\nu}}^-$. All the stoichiometry matrices are of size $N \times M$. All elements of $\boldsymbol{\nu}^+$ and $\boldsymbol{\nu}^-$ are non-negative whereas those of $\boldsymbol{\nu}$ can be negative, zero or positive.
From the stoichiometry matrix, we can directly compute the conservation relations in the reaction network as its left null space. The **left null space** *of* $\boldsymbol{\nu}$ is defined as the space of all vectors $\boldsymbol{l}$ for which

$$\boldsymbol{\nu}^{\mathrm{T}} \boldsymbol{l} = \mathbf{0}. \tag{1.3}$$

The left null space thus defines the conservation relations among the species in the reaction network.

## 1.3 Examples

### 1.3.1 Cyclic chain model

Consider the following cyclic chain reaction network with $N$ species and $M = N$ reactions:

$$\begin{array}{rcll} \mathrm{S}_i & \rightarrow & \mathrm{S}_{i+1}, & i = 1, \ldots, N-1 \\ \mathrm{S}_N & \rightarrow & \mathrm{S}_1. & \end{array} \tag{1.4}$$

For $N = 3$ the reaction network is

$$\begin{array}{rrcl} \text{Reaction } 1: & \mathrm{S}_1 & \rightarrow & \mathrm{S}_2 \\ \text{Reaction } 2: & \mathrm{S}_2 & \rightarrow & \mathrm{S}_3 \\ \text{Reaction } 3: & \mathrm{S}_3 & \rightarrow & \mathrm{S}_1. \end{array} \tag{1.5}$$

The stoichiometry matrices for this reaction network are:

$$\boldsymbol{\nu}^- = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.6}$$

$$\boldsymbol{\nu}^+ = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \tag{1.7}$$

and

$$\boldsymbol{\nu} = \boldsymbol{\nu}^+ - \boldsymbol{\nu}^- = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}. \tag{1.8}$$

The left null space $\boldsymbol{l}$ of $\boldsymbol{\nu}$ is

$$\boldsymbol{l} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{1.9}$$

which implies:

$$S_1 + S_2 + S_3 = \text{constant}. \tag{1.10}$$

This is the conservation relation enforced by the reactions in Eq. 1.5 at all times (in stationary and non-stationary state). For very large reaction networks, this relation may not be obvious and the left null space useful.

### 1.3.2 Colloidal aggregation model

Consider the following colloidal aggregation reaction network with $N$ species and $M = \left\lfloor \frac{N^2}{4} \right\rfloor$ reactions:

$$\mathrm{S}_i + \mathrm{S}_j \quad \rightarrow \quad \mathrm{S}_{i+j} \qquad i + j = N. \tag{1.11}$$

Species $S_i$ can be considered a multimer consisting of $i$ monomers.
For $N = 4$ the reaction network is

$$\begin{array}{lrcl} \text{Reaction 1}: & S_1 + S_1 & \rightarrow & S_2 \\ \text{Reaction 2}: & S_1 + S_2 & \rightarrow & S_3 \\ \text{Reaction 3}: & S_1 + S_3 & \rightarrow & S_4 \\ \text{Reaction 4}: & S_2 + S_2 & \rightarrow & S_4. \end{array} \tag{1.12}$$

The stoichiometry matrices for this reaction network are:

$$\boldsymbol{\nu}^- = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{1.13}$$

$$\boldsymbol{\nu}^+ = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \tag{1.14}$$

and

$$\boldsymbol{\nu} = \boldsymbol{\nu}^+ - \boldsymbol{\nu}^- = \begin{bmatrix} -2 & -1 & -1 & 0 \\ 1 & -1 & 0 & -2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{1.15}$$

The left null space $\boldsymbol{l}$ of $\boldsymbol{\nu}$ is

$$l = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \tag{1.16}$$

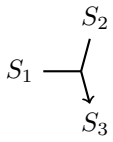
implying that

$$S_1 + 2S_2 + 3S_3 + 4S_4 = \text{constant}. \tag{1.17}$$

This is the conservation relation enforced by the reactions in Eq. 1.12 at all times (in stationary and non-stationary state). This means that the total number of bound and free monomers in the system is constant.

## 1.4 Graphical representation of reaction networks

Reaction networks can be represented in graphical form as *species dependency graph* or as *reaction dependency graph*. The species dependency graph is a directed hypegraph where each species is a node and directed hyper-edges represent reactions. For example for the reaction $S_1 + S_2 \rightarrow S_3$:




The species dependency graph depicts how species can be interconverted by reactions. It is a hypergraph, since every edge can have multiple origin nodes and multiple target nodes.

The dual representation is the *reaction dependency graph* [1]. The reaction dependency graph of a reaction network is a directed graph with nodes representing reactions and directed edges indicating the couplings between the reactions of the network. A directed edge is drawn from node $p$ to node $q$ if any of the reactants or products of reaction $p$ are involved as a reactant in reaction $q$ (Fig. 1.1). More formally, a directed edge is drawn from node $p$ to node $q$ if the vector obtained by performing

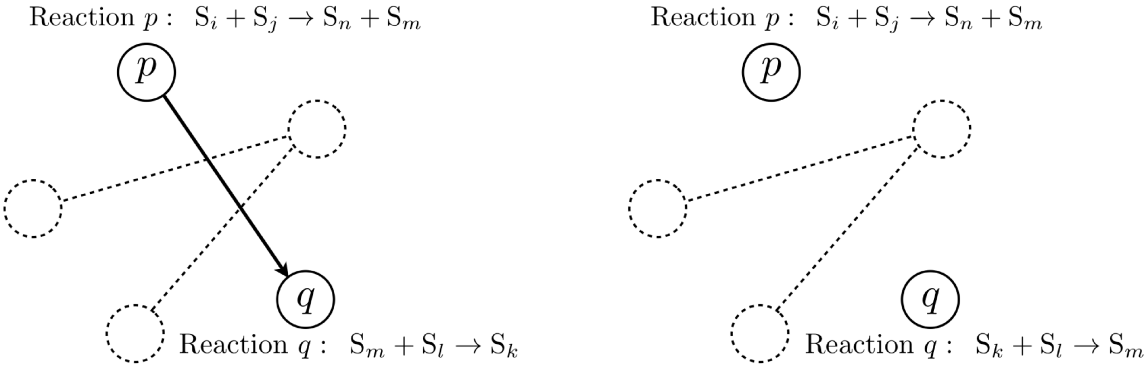



Figure 1.1: Illustration of the reaction dependency graph representation of a chemical reaction network. The nodes in a reaction dependency graph represent reactions. The directed edges (arrows) between the nodes represent the dependencies or the couplings between the reactions in the network. In the left panel, a directed edge is drawn from node $p$ to node $q$ because species $\mathrm{S}_m$ is a product of reaction $p$ and a reactant in reaction $q$. In the right panel, no edge is drawn from node $p$ to node $q$ since none of reactants or products of reaction $p$ are involved as a reactant in reaction $q$. In both panels, no directed edge is drawn from node $q$ to node $p$.

an element-wise logical *AND* operation between the binarized overall stoichiometry vector $\hat{\boldsymbol{\nu}}_p$ and the binarized reactant stoichiometry vector $\hat{\boldsymbol{\nu}}_q^-$ contains non-zero elements. Binarization is defined as:

$$\begin{aligned} \hat{\nu}_{i,p} &= 0 \text{ if } \quad \nu_{i,p} = 0 \\ \hat{\nu}_{i,p} &= 1 \text{ if } \quad \nu_{i,p} \neq 0 \end{aligned} \tag{1.18}$$

and

$$\begin{aligned} \hat{\nu}^-_{i,q} &= 0 \text{ if } \quad \nu^-_{i,q} = 0 \\ \hat{\nu}^-_{i,q} &= 1 \text{ if } \quad \nu^-_{i,q} > 0. \end{aligned} \tag{1.19}$$

## 1.5 Degree of coupling of a reaction network

The out-degree $d_\mu$ of a node $\mu$ in a directed graph is the number of directed edges leaving that node. In a reaction dependency graph of a chemical reaction network, it indicates the number of reactions that are influenced upon firing of the reaction represented by that node. We define the *degree of coupling* $d_c$ of a reaction network as the maximum out-degree of its reaction dependency graph, i.e.,

$$d_c = \max\{d_1, \ldots, d_M\}. \tag{1.20}$$

The degree of coupling $d_c$ is equal to the maximum number of reactions that are influenced by the firing of any reaction. The out-degree $d_\mu$ of reaction $\mu$ in the general chemical reaction network (Eq. 1.1) is

$$d_\mu = \sum_{\mu'=1}^{M} \frac{\sum_{i=1}^{N}(\hat{\nu}_{i,\mu} \wedge \hat{\nu}^{-}_{i,\mu'})}{\delta\left(\sum_{i=1}^{N}(\hat{\nu}_{i,\mu} \wedge \hat{\nu}^{-}_{i,\mu'})\right) + \sum_{i=1}^{N}(\hat{\nu}_{i,\mu} \wedge \hat{\nu}^{-}_{i,\mu'})} \tag{1.21}$$

where $\delta(\cdot)$ is the unit impulse function (the Kronecker delta symbol).

## 1.6 Classification of reaction networks

Based on their *degree of coupling*, we classify reaction networks into *weakly coupled* and *strongly coupled* networks. Weakly coupled networks have a degree of coupling $d_c$ that is bounded by a constant with increasing network size. Formally, in weakly coupled networks $d_c$ is $O(1)$. Strongly coupled networks are those where $d_c$ increases with network size, i.e., $d_c \notin O(1)$ and typically $O(N)$ or $O(M)$.

## 1.7 Examples

### 1.7.1 Cyclic chain model

The binarized stoichiometry matrices are

$$\hat{\boldsymbol{\nu}}^{-} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.22}$$

$$\hat{\boldsymbol{\nu}}^{+} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{1.23}$$

and

$$\hat{\boldsymbol{\nu}} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \tag{1.24}$$

The reaction dependency graph of this reaction network is shown in Fig. 1.2A. A directed edge is drawn from node 1 to node 2 since species $S_2$ is a product of reaction 1 and a reactant in reaction 2. The drawing of this directed edge can also be decided by performing the elementwise logical-and operation between $\hat{\boldsymbol{\nu}}_1$ and $\hat{\boldsymbol{\nu}}_2^-$:

$$\hat{\boldsymbol{\nu}}_1 \wedge \hat{\boldsymbol{\nu}}_2^- = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{1.25}$$

Since at least one of the elements in the resulting vector is non-zero, we draw a directed edge from node 1 to node 2. In addition, a directed edge is drawn from node 1 to itself, due to the presence of species 1. These two directed edges are the only edges originating from node 1. In a similar manner, directed edges are drawn from nodes 2 and 3 to complete the dependency graph as shown in Fig. 1.2A.

The degree of coupling of this network can be computed from Eqs. 1.20 and 1.21: Visual inspection of the dependency graph in Fig. 1.2A shows that the out-degree $d_\mu = 2$ for every $\mu$. Hence, the degree of the coupling is $d_\mathrm{c} = 2$.

This is the degree of coupling for the cyclic chain model with $N = 3$ species. The dependency graph for the same model with $N = 4$ species is shown in Fig. 1.2B, and we observe that also here $d_\mathrm{c} = 2$. In general, for the cyclic chain model with $N$ species (Eq. 1.4), $d_\mathrm{c} = 2$ independent of the size of the network. Therefore, the cyclic chain model is an example of a *weakly coupled* reaction network.

### 1.7.2 Colloidal aggregation model

The binarized stoichiometry matrices are

$$\hat{\boldsymbol{\nu}}^- = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{1.26}$$

$$\hat{\boldsymbol{\nu}}^+ = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{1.27}$$

and

$$\hat{\boldsymbol{\nu}} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{1.28}$$

The reaction dependency graph of this reaction network is shown in Fig. 1.2C. The degree of coupling can be computed from Eqs. 1.20 and 1.21. Visual inspection of the reaction dependency graph in Fig. 1.2C shows that the out-degrees of the nodes are $d_1 = 4$, $d_2 = 4$, $d_3 = 3$ and $d_4 = 3$. Hence, the degree of coupling is $d_\mathrm{c} = 4$ (Eq. 1.21).

**Classification:** This is the degree of coupling for the colloidal aggregation model with $N = 4$ species. The reaction dependency graph for the same model with $N = 5$ species is shown in Fig. 1.2D, and we observe that then $d_c = 6$. In general, for the aggregation model with $N$ species (Eq. 1.11), $d_c = 2N - 4$. Since $d_c$ increases with network size, $d_c \in O(N) \notin O(1)$. Therefore, the colloidal aggregation model is a *strongly coupled* reaction network.

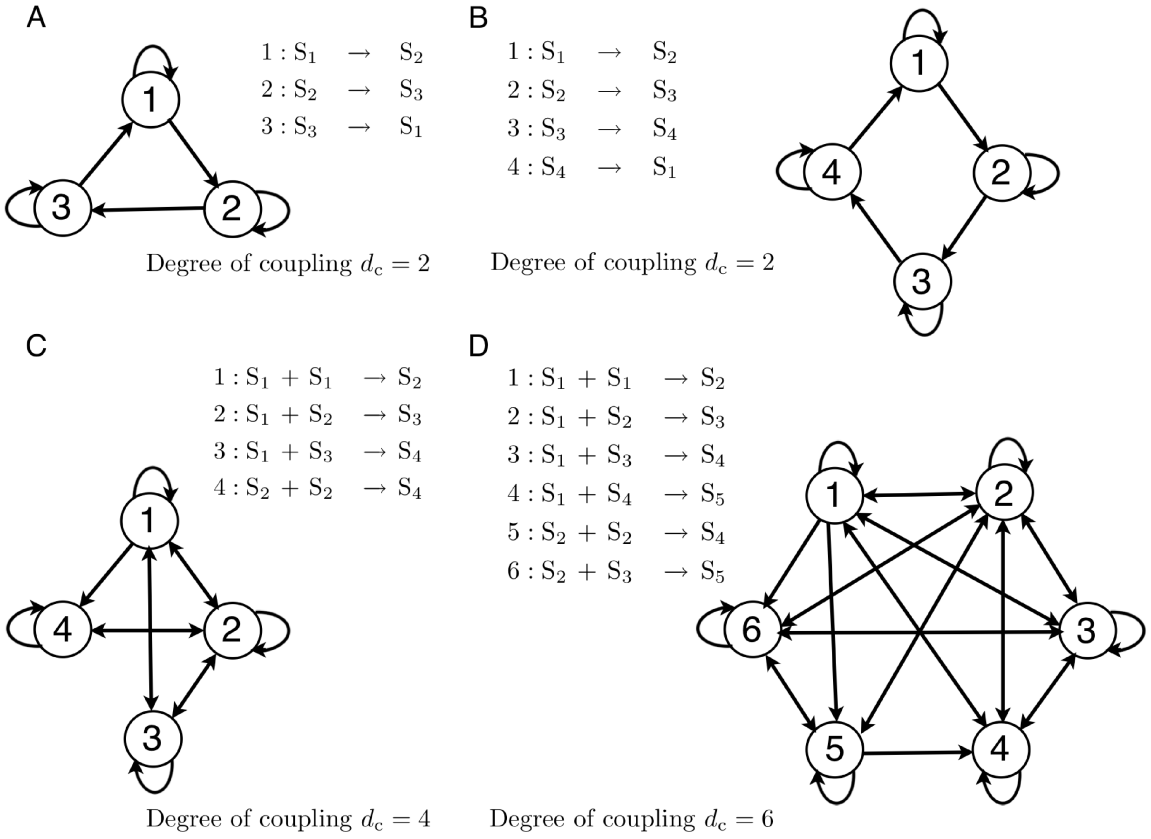



Figure 1.2: Reaction dependency graph representations of four example reaction networks. (A) and (B) show the reaction dependency graph of the cyclic chain model (Eq. 1.4) with $N = 3$ and 4 species, respectively. (C) and (D) show the reaction dependency graph of the colloidal aggregation model (Eq. 1.11) for $N = 4$ and 5, respectively.

# 2

# Microscopic Modeling and Simulation of Reaction Networks

*In this chapter:*

- The Chemical Master Equation description of reaction networks
- Kinetic Monte Carlo simulations
- Exact stochastic simulation algorithms
- Accelerated exact stochastic simulation algorithms
- The concept of partial propensities and partial-propensity stochastic simulation algorithms
- Exploiting symmetry in interactions

*Learning goals:*

- Be able to explain and understand the chemical master equation
- Know the physical basis of the chemical master equation
- Be able to implement a kinetic Monte Carlo simulation for a given reaction network
- Know at least two formulations of the exact SSA and their computational complexity
- Be able to implement exact SSA simulations
- Be able to write partial propensities for a given reaction network
- Know the concept of composition-rejection sampling

- Know how partial-propensity SSA works and what its computational cost it
- Be able to implement at least one formulation of the partial-propensity SSA

In the microscopic description of a chemical reaction networks, we explicitly track every molecule. That is, we explicitly track the copy number (population) of all molecules in the system. The molecules move around randomly according to thermal Brownian motion. They collide with each other at random, and a collision *may* lead to a reaction. Not every collision actually leads to a reaction, though. The collision energy and the orientation of the molecules need to be right. This is modeled by an intrinsic probability that the reaction occurs, given a collision has occurred.

## 2.1 The chemical master equation (CME)

In general, a system of chemical reactions can be considered to comprise $N$ species and $M$ reactions, such that

$$\sum_{i=1}^{N} \nu_{i,\mu}^{-} \mathrm{S}_i \xrightarrow{k_\mu} \sum_{i=1}^{N} \nu_{i,\mu}^{+} \mathrm{S}_i, \quad \mu = 1, \ldots, M. \tag{2.1}$$

Here, $\boldsymbol{\nu}^{-} = [\nu_{i,\mu}^{-}]$ and $\boldsymbol{\nu}^{+} = [\nu_{i,\mu}^{+}]$ are the stoichiometry matrices of the reactants and products, respectively. Both of these matrices are of size $N \times M$. The overall stoichiometry matrix $\boldsymbol{\nu}$ of the reaction network is given by $\boldsymbol{\nu} = \boldsymbol{\nu}^{+} - \boldsymbol{\nu}^{-}$. The elements of $\boldsymbol{\nu}^{-}$ and $\boldsymbol{\nu}^{+}$ are non-negative integers, while those of $\boldsymbol{\nu}$ can be positive, negative or zero. We denote by $\boldsymbol{\nu}_\mu$ the $\mu^{\text{th}}$ column of $\boldsymbol{\nu}$. $\mathrm{S}_i$ is the $i$-th species in the reaction network and $n_i(t)$ is its population (molecular copy number) at time $t$. The population vector $\boldsymbol{n}(t) = [n_1, \ldots, n_N]^{\mathrm{T}}(t)$ is the state of the system. The reactions occur in a reactor of volume $\Omega$, and the macroscopic reaction rate of reaction $\mu$ is $k_\mu$. See Fig. 2.1 for an illustration.
It is custom to define a variable $c_\mu$, called the specific probability rate of reaction $\mu$ [2, 3], such that

$$\begin{aligned} c_\mu \mathrm{d}t \quad = \quad & \text{Probability of reaction } \mu \text{ firing in the next infinitesimal time interval} \\ & [t, t + \mathrm{d}t) \text{ with randomly selected } \nu_{i,\mu}^{-} \text{ molecules of } \mathrm{S}_i \quad i = 1, \ldots, N. \end{aligned} \tag{2.2}$$

This makes $c_\mu$ the smallest non-zero probability rate of reaction $\mu$. It is the probability that *one* randomly selected combination of reactant molecules of reaction $\mu$ reacts in the next infinitesimal time interval $\mathrm{d}t$. The specific probability rate $c_\mu$ is related to the macroscopic reaction rate $k_\mu$ [2] as

$$c_\mu = \frac{k_\mu \left( \prod_{i=1}^{N} \nu_{i,\mu}^{-}! \right)}{\Omega^{\left(\sum_{i=1}^{N} \nu_{i,\mu}^{-}\right)-1}}. \tag{2.3}$$

The reaction degeneracy $h_\mu$ of reaction $\mu$ is defined as [2, 3]:

$$\begin{aligned} h_\mu(\boldsymbol{n}) \quad = \quad & \text{Number of distinct combinations by which the reactants of reaction } \mu \text{ can} \\ & \text{react to form products.} \end{aligned} \tag{2.4}$$

Therefore

$$h_\mu = \prod_{i=1}^{N} \binom{n_i}{\nu_{i,\mu}^{-}}, \tag{2.5}$$

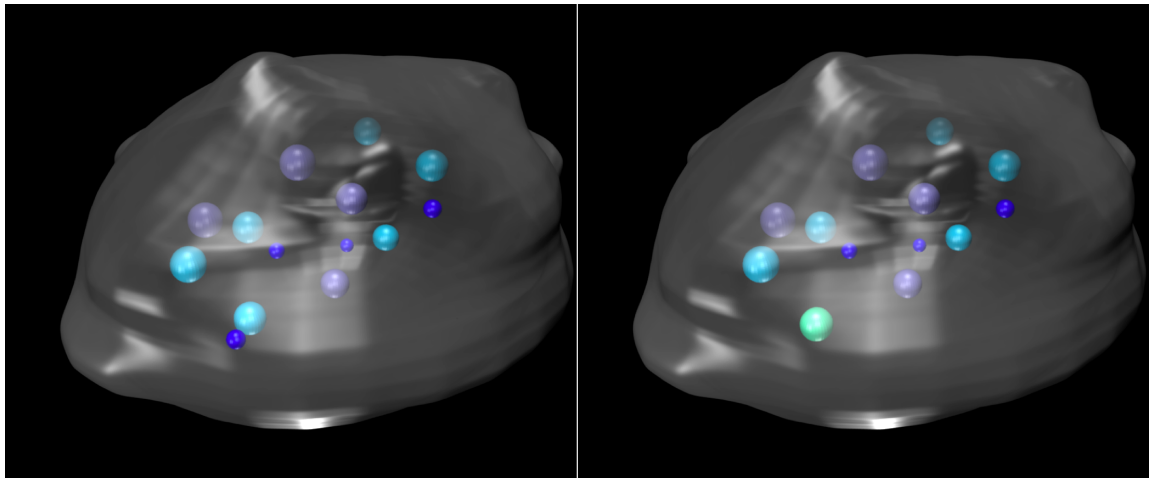


Figure 2.1: Illustration of molecules in a reactor. The reactor boundary is represented by the gray surface enclosing the molecules. The color of the molecules is used to encode the species. The molecules in the reactor posses kinetic energy and move around randomly. Upon collision they *may* react to form molecules of a different species. Single molecules can also get transformed into a new species, or decay. The left panel of shows a reactor with two molecules of different species in contact in the lower left corner. The right panel shows the formation of a molecule of a new species appearing in the place of the two molecules that are in contact in the left panel. This illustrates the occurrence of a reaction event.

where

$$\begin{pmatrix} n_i \\ \nu^-_{i,\mu} \end{pmatrix} = \frac{n_i!}{\nu^-_{i,\mu}!(n_i - \nu^-_{i,\mu})!} = \frac{(n_i - \nu^-_{i,\mu} + 1)(n_i - \nu^-_{i,\mu} + 2)\dots(n_i - 1)n_i}{\nu^-_{i,\mu}!}. \tag{2.6}$$

From Eqs. 2.2 and 2.4 the probability rate or the propensity $a_\mu$ of reaction $\mu$ becomes

$$\begin{aligned} a_\mu(\boldsymbol{n})\mathrm{d}t = c_\mu h_\mu(\boldsymbol{n})\mathrm{d}t \quad &= \quad \text{Probability of reaction } \mu \text{ firing in the next infinitesimal time} \\ &\quad \text{interval } [t, t+\mathrm{d}t) \text{ with } n_i \text{ molecules of } \mathrm{S}_i, \quad i = 1, \dots, N. \end{aligned} \tag{2.7}$$

Note that $a_\mu = c_\mu$ when $\boldsymbol{n} = \boldsymbol{\nu}^-_\mu$. This is hence the probability that the reaction $\mu$ fires within the next infinitesimal time interval using *any* of the possible reactant combinations.
The state probability distribution function is defined as

$$\begin{aligned} P(\boldsymbol{n}, t \mid \boldsymbol{n}_0, t_0) \quad &= \quad \text{Probability that the population of species is } \boldsymbol{n} \text{ at time } t, \\ &\quad \text{given a population } \boldsymbol{n}_0 \text{ at time } t_0. \end{aligned} \tag{2.8}$$

Using the Chapman-Kolmogorov equation [4, 5, 6, 7] describing the time evolution of $P$ for a Markov process

$$P(\boldsymbol{n}, t+\mathrm{d}t \mid \boldsymbol{n}_0, t_0) \quad = \quad \sum_{\boldsymbol{n}_1} P(\boldsymbol{n}, t+\mathrm{d}t \mid \boldsymbol{n}_1, t)\ P(\boldsymbol{n}_1, t \mid \boldsymbol{n}_0, t_0), \tag{2.9}$$

$$t_0 \le t \le t + \mathrm{d}t. \tag{2.10}$$

For the chemical reaction defined in Eq. 2.1 the above equation can be equivalently written as

$$\begin{aligned} P(\boldsymbol{n}, t+\mathrm{d}t \mid \boldsymbol{n}_0, t_0) &= \sum_{\mu=1}^{M} P(\boldsymbol{n}, t+\mathrm{d}t \mid \boldsymbol{n}-\boldsymbol{\nu}_\mu, t)\ P(\boldsymbol{n}-\boldsymbol{\nu}_\mu, t \mid \boldsymbol{n}_0, t_0) \\ &+ P(\boldsymbol{n}, t+\mathrm{d}t \mid \boldsymbol{n}, t)\ P(\boldsymbol{n}, t \mid \boldsymbol{n}_0, t_0), \end{aligned} \tag{2.11}$$

where $\boldsymbol{\nu}_\mu$ is the vector of stoichiometries of reaction $\mu$, and $P(\boldsymbol{n}, t+\mathrm{d}t \mid \boldsymbol{n}-\boldsymbol{\nu}_\mu, t)$ is the probability distribution $\Pi(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\, \boldsymbol{n}, t)$ of the Markov propagator such that the equation of motion for the states (population) can be written as

$$\boldsymbol{n}(t+\mathrm{d}t) = \boldsymbol{n}(t) + \boldsymbol{\Xi}(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\, \boldsymbol{n}, t), \tag{2.12}$$

where $\boldsymbol{\Xi}(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\, \boldsymbol{n}, t)$ is a random variable, called the Markov propagator, distributed such that $\boldsymbol{\Xi}(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\, \boldsymbol{n}, t) \sim \Pi(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\, \boldsymbol{n}, t)$.
Simplifying the notation, Eq. 2.11 reads

$$\begin{aligned} & P(\boldsymbol{n}, t+\mathrm{d}t) \\ = & \sum_{\mu=1}^{M} P(\boldsymbol{n}-\boldsymbol{\nu}_\mu, t)\ \text{Prob}\{\text{Reaction } \mu \text{ fires once in } [t, t+\mathrm{d}t) \text{ given } \boldsymbol{n}-\boldsymbol{\nu}_\mu \text{ at } t\} \qquad (2.13) \\ & + P(\boldsymbol{n}, t)\ \text{Prob}\{\text{No reaction in } [t, t+\mathrm{d}t) \text{ given } \boldsymbol{n} \text{ at } t\}. \qquad (2.14) \end{aligned}$$

The reasoning behind this equation is as follows: The first summand on the right-hand side is the probability that the chemical reaction system moves from a population state $\boldsymbol{n}-\boldsymbol{\nu}_\mu$ at time $t$ to the population state $\boldsymbol{n}$ at time $t+\mathrm{d}t$. The second term is the probability that the system is at state $\boldsymbol{n}$ at time $t$ and stays at the same state at time $t+\mathrm{d}t$. The sum of these two terms is therefore the probability of the system being in state $\boldsymbol{n}$ at time $t+\mathrm{d}t$.
The probability of reaction $\mu$ firing once in the time interval $[t, t+\mathrm{d}t)$ is the probability that any one combination of its reactant molecules reacts, times the probability that none of the other combinations react. According to Eq. 2.2, the probability for a *particular* reactant combination to react is $c_\mu \mathrm{d}t$. The probability for any of the remaining $h_\mu - 1$ combinations *not* to react is $(1-c_\mu \mathrm{d}t)^{h_\mu - 1}$. Therefore, the probability for *this particular* combination of molecules to react, but none of the others, is $c_\mu \mathrm{d}t(1-c_\mu \mathrm{d}t)^{h_\mu-1}$. Since this can equally happen with any of the $h_\mu$ possible reactant combinations, the probability for *any one* of the molecular combinations to react, but no other, is:

$$\begin{aligned} \text{Prob}\{\text{Reaction } \mu \text{ fires once in } [t, t+\mathrm{d}t) \text{ given } \boldsymbol{n} \text{ at } t\} &= P(\boldsymbol{n}+\boldsymbol{\nu}_\mu, t+\mathrm{d}t \mid \boldsymbol{n}, t) \\ &= \Pi(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\, \boldsymbol{n}, t) \\ &= h_\mu c_\mu \mathrm{d}t(1-c_\mu \mathrm{d}t)^{h_\mu(\boldsymbol{n})-1} \\ &= c_\mu h_\mu(\boldsymbol{n})\mathrm{d}t + O(\mathrm{d}t^2) \\ &= a_\mu(\boldsymbol{n})\mathrm{d}t + O(\mathrm{d}t^2), \end{aligned} \tag{2.15}$$

where $O(\mathrm{d}t^2)$ is the Bachmann-Landau big-$O$ symbol* such that $\lim_{\mathrm{d}t\to 0} \frac{O(\mathrm{d}t^\alpha)}{\mathrm{d}t} = 0$ for $\alpha > 1$. The expansion in $\mathrm{d}t$ makes use of the binomial series $(1+x)^n = 1 + nx + \frac{1}{2}n(n-1)x^2 + \ldots$ for $x = -c_\mu \mathrm{d}t$ and $n = h - 1$.

*$f(x) \in O(g(x))$, or $f(x)$ is $O(g(x))$, implies that $f(x)$ is asymptotically bounded from above by $g(x)$, i.e., $\exists \epsilon > 0,\ x_0 : \forall x > x_0,\ |f(x) \leq \epsilon g(x)|$

Extending the above result, the probability that no reactions occur in $[t, t+\mathrm{d}t)$ is given by

$$\begin{aligned}\text{Prob}\{\text{No reaction in } [t,t+\mathrm{d}t) \text{ given } \boldsymbol{n} \text{ at } t\} &= P(\boldsymbol{n}, t+\mathrm{d}t \mid \boldsymbol{n}, t) \\ &= 1-\sum_{\mu=1}^{M} a_\mu(\boldsymbol{n})\mathrm{d}t + O(\mathrm{d}t^2) \\ &= 1-a(\boldsymbol{n})\mathrm{d}t + O(\mathrm{d}t^2), \end{aligned} \tag{2.16}$$

where the total propensity $a(\boldsymbol{n}) = \sum_{\mu=1}^{M} a_\mu(\boldsymbol{n})$. Equation 2.14 can therefore be written as

$$\begin{aligned} P(\boldsymbol{n}, t+\mathrm{d}t) &= \sum_{\mu=1}^{M} P(\boldsymbol{n}-\boldsymbol{\nu}_\mu, t)\ \left[a_\mu(\boldsymbol{n}-\boldsymbol{\nu}_\mu)\mathrm{d}t + O(\mathrm{d}t^2)\right] \\ &\quad + P(\boldsymbol{n}, t)\ \left[1-\sum_{1}^{M} a_\mu(\boldsymbol{n})\mathrm{d}t + O(\mathrm{d}t^2)\right], \end{aligned} \tag{2.17}$$

i.e.,

$$\begin{aligned} \frac{P(\boldsymbol{n}, t+\mathrm{d}t) - P(\boldsymbol{n}, t)}{\mathrm{d}t} &= \sum_{\mu=1}^{M} \left[a_\mu(\boldsymbol{n}-\boldsymbol{\nu}_\mu) + \frac{O(\mathrm{d}t^2)}{\mathrm{d}t}\right]\ P(\boldsymbol{n}-\boldsymbol{\nu}_\mu, t) \\ &\quad - P(\boldsymbol{n}, t)\ \left[\sum_{\mu=1}^{M} a_\mu(\boldsymbol{n}) + \frac{O(\mathrm{d}t^2)}{\mathrm{d}t}\right]. \end{aligned} \tag{2.18}$$

Taking the limit $\lim_{\mathrm{d}t\to 0}$ we obtain the *chemical master equation* (CME) [8, 3] as

$$\frac{\partial P(\boldsymbol{n}, t)}{\partial t} = \sum_{\mu=1}^{M} a_\mu(\boldsymbol{n}-\boldsymbol{\nu}_\mu)\ P(\boldsymbol{n}-\boldsymbol{\nu}_\mu, t) - \ P(\boldsymbol{n}, t)\ a(\boldsymbol{n}). \tag{2.19}$$

The CME is the differential equation governing the evolution of the state probability density function $P(\boldsymbol{n}, t)$. The CME is the forward-time equation of a discrete-state jump Markov process obtained as a direct consequence of the Chapman-Kolmogorov equation [4, 5, 6, 7] (Eq. 2.11). The CME can also be written as

$$\frac{\partial P(\boldsymbol{n}, t)}{\partial t} = \sum_{\mu=1}^{M} \left(\mathbb{E}^{-\boldsymbol{\nu}_\mu} - 1\right) a_\mu(\boldsymbol{n}) P(\boldsymbol{n}, t), \tag{2.20}$$

where $\mathbb{E}$ is the step operator such that $\mathbb{E}^{\boldsymbol{a}} f(\boldsymbol{n}) = f(\boldsymbol{n}+\boldsymbol{a})$. The solution of the CME can therefore be written as

$$P(\boldsymbol{n}, t) = \mathrm{e}^{t\left[\sum_{\mu=1}^{M}\left(\mathbb{E}^{-\boldsymbol{\nu}_\mu}-1\right)a_\mu(\boldsymbol{n})\right]} P(\boldsymbol{n}, 0). \tag{2.21}$$

The initial condition is given by

$$P(\boldsymbol{n}, t=0) = \delta(\boldsymbol{n}-\boldsymbol{n}_0), \tag{2.22}$$

where $\boldsymbol{n}_0$ is the population at time $t=0$ and $\delta(\cdot)$ is the Kronecker delta or the unit impulse function.

## 2.2 Kinetic Monte Carlo: The stochastic simulation algorithm (SSA)

The CME provides an exact description of the kinetics of mesoscopic chemical reaction systems that are well stirred and thermally equilibrated [3]. The CME incorporates the effects of fluctuations due to low copy numbers of species by describing the effect of discrete nature of molecules involved in chemical reactions. At larger $\Omega$, when the population $\boldsymbol{n}$ increases, the effect of intrinsic noise becomes progressively smaller. As seen in Sec. 4.1, for very large reactor volumes $\Omega$ (or large population $\boldsymbol{n}$ under the assumption that $\boldsymbol{n}$ increases proportionally with $\Omega$), the CME reduces to the classical RRE (see Eq. 4.10). For intermediate values of $\Omega$, the CME leads to the chemical Kramer-Moyal equation and to the nonlinear Fokker-Planck equation (See Sec. 3.2.1).

Solving the CME, however, is riddled with problems. Analytically, the CME is generally intractable, except for networks of unimolecular reactions (i.e., when $\sum_{i=1}^{N} \nu_{i,\mu}^{-} \leq 1 \ \forall \mu$ in Eq. 2.1). Such networks are called *linear reaction networks* since the propensities are linear in the population. For networks with even a single bimolecular or higher order reaction (i.e., if $\sum_{i=1}^{N} \nu_{i,\mu}^{-} > 1$ for at least one reaction in Eq. 2.1), the solution of the CME is analytically not accessible, except in special cases. Such networks are called *nonlinear reaction networks* since at least one of the propensities is nonlinear in the population.

Numerically simulating the CME using, for example, finite differences is also infeasible for large networks due to the high dimensionality of the domain of the probability distribution $P(\boldsymbol{n}, t)$, which leads to an exponential increase in computational and memory cost with network size. These problems, however, can be circumvented using Gillespie's stochastic simulation algorithm (SSA), a kinetic Monte Carlo scheme [9, 10, 11, 12, 13, 2, 14, 3]. In SSA, the probability $P(\boldsymbol{n}, t)$ whose time evolution is given by the CME is replaced by the joint probability of a reaction event $p(\tau, \mu \mid \boldsymbol{n}(t))$, defined as

$$\begin{aligned} p(\tau, \mu \mid \boldsymbol{n}(t))\mathrm{d}\tau \quad &= \quad \text{Probability that the next reaction is } \mu \text{ and it fires in} \\ &\quad\ [t+\tau, t+\tau+\mathrm{d}\tau) \text{ given } \boldsymbol{n} \text{ at time } t. \end{aligned} \tag{2.23}$$

This probability $p$ is derived as follows: Consider that the time interval $[t, t+\tau]$ is divided into $k$ equal intervals of length $\frac{\tau}{k}$ plus a last interval $(k+1)$ of length $\mathrm{d}\tau$ (Fig. 2.2). The definition

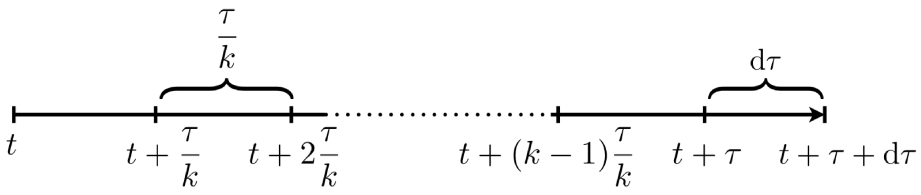



Figure 2.2: Division of the time interval $[t, t+\tau+\mathrm{d}\tau)$ into $k+1$ intervals. Here, $t$ represents the current time. The only reaction firing is reaction $\mu$ in the $(k+1)^{\text{th}}$ infinitesimally small time interval $[t+\tau, t+\tau+\mathrm{d}\tau)$.

(Eq. 2.23) of $p(\tau, \mu \mid \boldsymbol{n}(t))$ dictates that no reactions occur in each of the first $k$ intervals and that

reaction $\mu$ fires once in the last interval. Therefore, using Eqs. 2.15 and 2.16

$$p(\tau, \mu \mid \boldsymbol{n}(t))\mathrm{d}\tau \;=\; \left[1 - a(\boldsymbol{n})\frac{\tau}{k} + O\left(\frac{\tau^2}{k^2}\right)\right]^k \left[a_\mu(\boldsymbol{n})\mathrm{d}\tau + O\left(\mathrm{d}\tau^2\right)\right].$$

Dividing both sides of the equation by $\mathrm{d}\tau$ and taking the limit $\lim_{\mathrm{d}\tau\to 0}$ we obtain

$$p(\tau, \mu \mid \boldsymbol{n}(t)) \;=\; \left[1 - a(\boldsymbol{n})\frac{\tau}{k} + O\left(\frac{\tau^2}{k^2}\right)\right]^k a_\mu(\boldsymbol{n}).$$

Taking the limit $\lim_{k\to\infty}$, and noting that $\lim_{k\to\infty} O\left(\frac{\tau^2}{k^2}\right) = 0$, we get

$$p(\tau, \mu \mid \boldsymbol{n}(t)) \;=\; \mathrm{e}^{-a(\boldsymbol{n})\tau} a_\mu(\boldsymbol{n}). \tag{2.24}$$

Summing Eq. 2.24 over all reactions (summing over $\mu$) we get the marginal probability density function of $\tau$ as

$$\begin{aligned} p(\tau \mid \boldsymbol{n}(t)) \;&=\; \sum_{\mu=1}^{M} a_\mu(\boldsymbol{n})\mathrm{e}^{-a(\boldsymbol{n})\tau} \\ &=\; a(\boldsymbol{n})\mathrm{e}^{-a(\boldsymbol{n})\tau}. \end{aligned} \tag{2.25}$$

Similarly, integrating Eq. 2.24 over $\tau$'s we get the marginal probability distribution function of $\mu$ as

$$\begin{aligned} p(\mu \mid \boldsymbol{n}(t)) \;&=\; \int_0^\infty a_\mu(\boldsymbol{n})\mathrm{e}^{-a(\boldsymbol{n})\tau}\mathrm{d}\tau \\ &=\; \frac{a_\mu(\boldsymbol{n})}{a(\boldsymbol{n})}. \end{aligned} \tag{2.26}$$

From Eqs. 2.24, 2.25 and 2.26 we observe that

$$p(\tau, \mu \mid \boldsymbol{n}(t)) = p(\tau \mid \boldsymbol{n}(t))\, p(\mu \mid \boldsymbol{n}(t)) \tag{2.27}$$

thus inferring that $\mu$ and $\tau$ are statistically independent random variables.
Sampling from Eq. 2.24 constitutes a stochastic simulation of the underlying chemical reaction network. By sampling a reaction event and propagating the simulation in time according to Eq. 2.24, we obtain exact, time resolved trajectories of the population $\boldsymbol{n}$ as governed by the CME. The SSA, however, is a Monte Carlo scheme and hence several independent runs need to performed in order to obtain a stable representation of the probability function $P(\boldsymbol{n}, t)$. In Chapter **??** we describe the different implementations or formulations of the SSA used to simulate chemical kinetics, and we review their computational costs. Subsequently, in Chapter 2.4 we will present a more efficient class of SSA formulations called partial-propensity methods that use a novel quantity called *partial propensity* instead of the conventional reaction propensities.

## 2.3 Exact SSA

All formulations of Gillespie's SSA aim to simulate chemical kinetics by sampling the random variables $\tau$ (time to the next reaction) and $\mu$ (index of the next reaction) according to Eq. 2.24

and propagating the state of the chemical system in time. Since SSA is a Monte Carlo scheme, several independent runs of each simulation need to be performed in order to estimate the state (population) probability function $P(\boldsymbol{n}, t)$.
Existing SSA formulations can be classified into *exact* and *approximate* methods. Exact methods sample from the joint probability distribution in Eq. 2.24. These formulations include the first reaction method (FRM) [2], the direct method (DM) [14], Gibson-Bruck's next-reaction method (NRM) [1], a Gibson-Bruck variant of the DM [1], the optimized direct method (ODM) [15], the sorting direct method (SDM) [16], the logarithmic direct method (LDM) [17] and the composition-rejection formulation (SSA-CR) [18]. Approximate SSA formulations provide better computational efficiency for larger numbers of molecules by sampling from an approximation to the joint probability distribution in Eq. 2.24. These methods include $\tau$-leaping [19, 20, 21, 22], $k_\alpha$-leaping [19], $R$-leaping [23], $L$-leap [24], $K$-leap [25], the slow-scale method [26], and implicit $\tau$-leaping [27].
In the following, we focus on *exact* SSA formulations, but briefly also mention the fundamental idea, benefits and limitations of *approximate* SSAs towards the end of this chapter.
Exact SSA formulations sample $\mu$ and $\tau$ from Eq. 2.24, and only one reaction $\mu$ is executed per time step. The time step size $\tau$ is itself a random variable. The population state of the chemical system is subsequently updated. The fundamental steps in every exact SSA formulation are thus:

1. Sample $\tau$ and $\mu$ from Eq. 2.24.
2. Update population $\boldsymbol{n}$.
3. Recompute the reaction propensities $a_\mu$.

Various algorithms have been developed to perform these steps, and they constitute different SSA formulations with different computational costs. We quantify computational cost by the CPU time needed to execute one reaction event. The computational cost of any SSA formulation depends on the coupling class of the simulated reaction network. We provide here a review of various exact SSA formulations and analyze their computational costs as a function of the coupling class of the reaction network.

### 2.3.1 The first reaction method (FRM)

FRM [2] is one of the earliest exact SSA formulations. In this formulation, the time $\tau_\mu$ when reaction $\mu$ fires next is computed according to the probability function

$$p(\tau_\mu \mid \boldsymbol{n}(t)) = a_\mu \, \mathrm{e}^{-a_\mu \tau_\mu} \tag{2.28}$$

using the inversion method. Subsequently, the next reaction $\mu$ is chosen to be the one with the minimum $\tau_\mu$, and the time $\tau$ to the next reaction is set to the minimum $\tau_\mu$. The algorithm is given in Table 2.1.
The computational cost of FRM is $O(M)$ [1, 15, 28, 29] where $M$ is the number of reactions in the network. This is due to steps 2 and 4 (see Table 2.1), both of which have a runtime of $O(M)$: step 2 involves generating $M$ random numbers and step 4 involves recomputing all $M$ reaction propensities.

1. Set $t \leftarrow 0$; initialize $\boldsymbol{n}$, $a_\mu \, \forall \mu$, and $a$
2. Sample $\tau_\mu$ according to Eq. 2.28 for each reaction $\mu$: For each reaction generate an *i.i.d.* uniform random number $r_\mu \in [0, 1)$ and compute $\tau_\mu \leftarrow -a_\mu^{-1} \log(r_\mu)$. $\tau \leftarrow \min\{\tau_1, \ldots, \tau_M\}$
3. $\mu \leftarrow$ the index of minimum$\{\tau_1, \ldots, \tau_M\}$
4. Update: $\boldsymbol{n} \leftarrow \boldsymbol{n} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction $\mu$; recompute all $a_\mu$ and $a$
5. $t \leftarrow t + \tau$; go to step 2

Table 2.1: Algorithm for the first reaction method (FRM)

### 2.3.2 The direct method (DM)

DM [14] samples the next reaction index $\mu$ according to Eq. 2.26 using linear search over the reaction propensities. The time $\tau$ to the next reaction is sampled according to Eq. 2.25. The algorithm is given in Table 2.2.

1. Set $t \leftarrow 0$; initialize $\boldsymbol{n}$, $a_\mu \, \forall \mu$, and $a$
2. Sample $\mu$ using linear search according to Eq. 2.26: generate a uniform random number $r_1 \in [0, 1)$ and determine $\mu$ as the smallest integer satisfying $r_1 < \sum_{\mu'=1}^{\mu} a_{\mu'}/a$
3. Sample $\tau$ according to Eq. 2.25: generate a uniform random number $r_2 \in [0, 1)$ and compute $\tau$ as $\tau \leftarrow -a^{-1} \log(r_2)$
4. Update: $\boldsymbol{n} \leftarrow \boldsymbol{n} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction $\mu$; recompute all $a_\mu$ and $a$
5. $t \leftarrow t + \tau$; go to step 2

Table 2.2: Algorithm for the direct method (DM)

The computational cost of DM is also $O(M)$ [1, 15, 28, 29]. This is due to steps 2 and 4 in the algorithm (see Table 2.2), both of which have a worst-case runtime of $O(M)$. In terms of absolute runtimes, however, DM is more efficient that FRM since it does not involve the expensive step of generating $M$ random numbers for each reaction event.

### 2.3.3 Next reaction method (NRM)

NRM [1] is an improvement over FRM in which the $M - 1$ unused reaction times are suitably reused, and efficient data structures such as indexed minimum priority queues [1] and dependency graphs are introduced. The indexed priority queue is used to efficiently find the minimum among all $\tau_\mu$; the dependency graph is a data structure that stores for each reaction the indices of the propensities that have to be recomputed upon firing of this reaction (see Sec. **??**). This avoids

having to recompute all $a_\mu$ after every reaction event. The formulation of NRM is in terms of the absolute time to the next reaction $\tau^{\mathrm{g}}$ instead of the relative time to the next reaction $\tau$ that is used in DM and FRM. The algorithm of NRM is given in Table 2.3.

---

1. Set $t \leftarrow 0$; initialize $\boldsymbol{n}$, the dependency graph, $a_\mu \, \forall \mu$, and $a$
2. For each reaction $\mu$ generate a uniform random number $r_1 \in [0, 1)$ and compute as $\tau_\mu^{\mathrm{g}} \leftarrow -a_\mu^{-1} \log(r_1)$. Store these $\tau_\mu^{\mathrm{g}}$'s in a minimum priority queue, indexed by the reaction label
3. The reaction $\mu$ with the minimum $\tau_\mu^{\mathrm{g}}$ can be found at the top of the priority queue. Set $\tau^{\mathrm{g}} \leftarrow \tau_\mu^{\mathrm{g}}$
4. Update: $\boldsymbol{n} \leftarrow \boldsymbol{n} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction $\mu$
5. Set $t \leftarrow \tau^{\mathrm{g}}$
6. For all reactions $i$ whose propensities change upon firing of reaction $\mu$ (i.e., the destination nodes of all directed edges leaving node $\mu$ in the dependency graph):
   - 6.1. Recompute the propensity $a_i$. Store the old propensity in $a^{\mathrm{old}}$
   - 6.2. if $i \neq \mu$, set $\tau_i^{\mathrm{g}} \leftarrow t + \frac{a^{\mathrm{old}}}{a_i}(\tau_i^{\mathrm{g}} - t)$
   - 6.3 if $i = \mu$, set $\tau_i^{\mathrm{g}} \leftarrow t - a_i^{-1} \log(r_2)$, where $r_2$ is a uniform random number in $[0, 1)$
   - 6.4 Update the priority queue with the new value of $\tau_i^{\mathrm{g}}$ for reaction $i$
7. Go to step 3

---

Table 2.3: Algorithm for the next reaction method (NRM)

The data structures in NRM, together with the reuse of reaction times, reduce the computational cost to $O(d_{\mathrm{c}} \log_2 M)$, where $d_{\mathrm{c}}$ is the degree of coupling of the reaction network. This scaling of the computational cost is due to step 6 in Table 2.3. In this step, a maximum of $d_{\mathrm{c}}$ reaction times need to be recomputed. Additionally, every change involves ensuring that the tree structure of minimum priority queue has nodes that always carry smaller reaction times than their children. The computational cost of this update operation is $O(\log_2 M)$ and hence the overall cost of step 6 is $O(d_{\mathrm{c}} \log_2 M)$. For strongly coupled networks, $d_{\mathrm{c}}$ is a function of $M$ and is $O(M)$. The computational cost of NRM is thus $O(M \log_2 M)$ for strongly coupled networks. Even for some weakly coupled networks for which $d_{\mathrm{c}} \ll M$ and is $O(1)$, the computational cost of NRM has been empirically shown to be $O(M)$ [15]. This is due to the additional overhead, memory-access operations, and cache misses introduced by the complex data structures (indexed priority queue, dependency graph) of NRM. The absolute runtime of NRM is, however, still superior to that of FRM and DM.

### 2.3.4 Optimized direct method (ODM)

ODM [15] is an improvement over DM where the reactions are approximately sorted in descending order of firing frequency. Like DM, ODM uses linear search to sample the index of the next reaction

and the reordering makes it more probable to find the next reaction close to the beginning of the list, reducing the average search depth. ODM estimates the firing frequencies of all reactions during a short pre-simulation run of about 5–10% of the length of the entire simulation [15, 16]. In order to reduce the cost of updating the propensities after a reaction has fired, ODM also uses a dependency graph. The algorithm of ODM is similar to that of DM (see Table 2.2) except that the propensities of the reactions are stored in the descending order of their estimated firing frequency. Irrespective of the degree of coupling of the network, the computational cost of ODM is $O(M)$, which was also empirically confirmed in benchmarks [15]. The absolute runtimes of ODM, however, are smaller than those of DM and NRM [15], especially for multiscale (stiff) reaction networks whose propensities span several orders of magnitude. In stiff networks, few reactions fire much more frequently than all others, and having the frequent reactions at the top of the list greatly reduces the average search depth in the linear search.

### 2.3.5 Sorting direct method (SDM)

SDM [16] is a variant of ODM that does not use pre-simulation runs, but dynamically shifts up a reaction in the reaction list whenever it fires ("bubbling up" of the more frequent reactions). Like ODM, SDM is also especially efficient for multiscale reaction networks. Among multiscale reaction networks, the strategy of dynamically sorting the reactions is especially suited to deal with temporal changes in firing frequency that ODM fails to capture. The algorithm of SDM is given in Table 2.4.

The strategy of dynamically sorting the reactions reduces the prefactor of the computational cost of SDM compared to that of ODM, but the scaling remains $O(M)$ [16], irrespective of the degree of coupling of the network [28, 29].

1. Set $t \leftarrow 0$; initialize $\boldsymbol{n}$, $a_\mu \, \forall \mu$, $a$, the change to the total propensity $\Delta a \leftarrow 0$, the dependency graph and ordering list $\boldsymbol{l}$ such that $l_j = j$ for $j = 1, \ldots, M$.
2. Sample $\mu$ using linear search according to Eq. 2.26: generate a uniform random number $r_1 \in [0, 1)$ and determine $j$ as the smallest integer satisfying $r_1 < \sum_{j'=1}^{j} a_{l_{j'}}/a$. Set $\mu \leftarrow j$
3. If $j \neq 1$, swap $l_j$ and $l_{j-1}$
4. Sample $\tau$ according to Eq. 2.25: generate a uniform random number $r_2 \in [0, 1)$ and compute $\tau$ as $\tau \leftarrow -a^{-1} \log(r_2)$
5. Update: $\boldsymbol{n} \leftarrow \boldsymbol{n} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction $\mu$
6. For all reactions $i$ whose propensities change upon firing of reaction $\mu$ (i.e., the destination nodes of all directed edges leaving node $\mu$ in the dependency graph):
   6.1. Update $\Delta a \leftarrow \Delta a - a_i$
   6.2. Recompute the propensity $a_i$
   6.3. Update $\Delta a \leftarrow \Delta a + a_i$
7. Set $a \leftarrow a + \Delta a$
8. $t \leftarrow t + \tau$ and set $\Delta a \leftarrow 0$; go to step 2

Table 2.4: Algorithm for the sorting direct method (SDM)

### 2.3.6 Logarithmic direct method (LDM)

LDM [17] uses a binary search tree (recursive bisection) on an ordered linear list of cumulative sums of propensities $s_j = \sum_{\mu'=1}^{j} a_{\mu'}, \quad j = 1, \ldots, M$ and $s_0 = 0$, to find the next reaction $\mu$. The algorithm for LDM is given in Table 2.5.
The binary search tree to sample the next reaction index reduces the search depth to $O(\log_2 M)$. Irrespective of the degree of coupling, however, the update step (Step 6 in Table 2.5) is $O(M)$ since on average $(M + 1)/2$ and in the worst case $M$ cumulative sums of propensities need to be recomputed, rendering the computational cost of LDM $O(M)$.

1. Set $t \leftarrow 0$; initialize $\boldsymbol{n}$, $a_\mu \, \forall \mu$, the dependency graph and the partial sums $s_j$
2. Sample $\mu$ using recursive bisection search according to Eq. 2.26: generate a uniform random number $r_1 \in [0, 1)$ and perform binary search until $s_{\mu-1} \leq s_M r_1 = a r_1 < s_\mu$ where $a = s_M$ is the total propensity of all reactions
3. Sample $\tau$ according to Eq. 2.25: generate a uniform random number $r_2 \in [0, 1)$ and compute $\tau$ as $\tau \leftarrow -a^{-1} \log(r_2)$
4. Update: $\boldsymbol{n} \leftarrow \boldsymbol{n} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction $\mu$
5. Using the dependency graph, find the smallest reaction index $i$ whose propensity is affected by $\mu$
6. for $j = i \ldots M$
   6.1. Recompute $s_j$
7. $t \leftarrow t + \tau$; go to step 2

Table 2.5: Algorithm for the logarithmic direct method (LDM)

### 2.3.7 Composition-rejection method (SSA-CR)

SSA-CR [18] uses composition-rejection sampling to determine the index of the next reaction. Composition-rejection sampling [30, 18, 29] is a way of sampling realizations of a random variable according to a given probability function. In SSA, the discrete probability function to sample the next reaction index $\mu$ is $p(\mu \mid \boldsymbol{n}(t))$ (see Eq. 2.26). The sampling process starts by binning the propensities $a_i$ according to their value, and then proceeds in two steps: The composition step is used to identify the bin by linear search, the rejection step is used to identify the $a_\mu$, and hence the next reaction index $\mu$, inside that bin.
In SSA-CR, the propensities $a_i$ are sorted into $G_\mathrm{a} = \log_2(a_\mathrm{max}/a_\mathrm{min}) + 1$ bins such that bin $b$ contains all $a_i$ where $b$ is determined by the condition: $2^{b-1} a_\mathrm{min} \leq a_i < 2^b a_\mathrm{min}$. The constants $a_\mathrm{min}$ and $a_\mathrm{max}$ are the smallest non-zero and largest value that any of the $a_i$'s can assume during the simulation. The value of $a_\mathrm{min}$ is given by the minimum specific probability rate among all reactions in the network. The value of $a_\mathrm{max}$ can be estimated by using physical reasoning. In cases where such an estimation is not possible, the number of bins $G_\mathrm{a}$ can also be dynamically increased during the simulation.
In SSA-CR, sampling the next reaction index $\mu$ proceeds in two steps: (1) composition sampling using linear search over the $G_\mathrm{a}$ bins and a uniform random number in $[0, 1)$ to identify the bin $b$ such that

$$b = \min \left[ b' \; : \; r_1 a < \sum_{i=1}^{b'} \alpha_i \right] , \tag{2.29}$$

where the total propensity $\alpha_b$ of each bin is computed by summing up the $a_i$'s in bin $b$, and

subsequently (2) rejection (also known as acceptance-rejection) sampling to identify the reaction index $\mu$ in that bin $b$. This is done by generating a uniformly distributed random number $r_2$ in $[0, 2^b a_{\min})$ and a uniformly distributed random integer $r_3$ between 1 and the number of elements in bin $b$. If the $r_3^{\text{th}}$ element in bin $b$ is greater than or equal to $r_2$, the corresponding reaction index is chosen as the index of the next reaction. If the inequality is not satisfied, the rejection step is repeated. This procedure is illustrated in Fig. 2.3. Assume that the composition step has chosen bin 2 as the bin containing the next reaction index. The rejection step then samples uniformly random points inside the rectangle defining the value range of this bin (bold rectangle). A sample is accepted if it falls inside one of the shaded bars representing the propensities. If the first sample (point A in Fig. 2.3 with $r_3 = 3$ and $r_2 > a_2$) is rejected, sampling is repeated until the point falls inside one of the bars (point B in Fig. 2.3 with $r_3 = 2$ and $r_2 < a_4$). By binning the propensities as described above, one ensures that they cover at least 50% of each bin's total area, ensuring that the average number of rejection steps needed is less than or equal to two. Once a reaction is executed, the affected propensities are updated and their bin memberships are reassigned. Reassigning a bin membership can be achieved in $O(1)$ time. The algorithm for SSA-CR is given in Table 2.6.

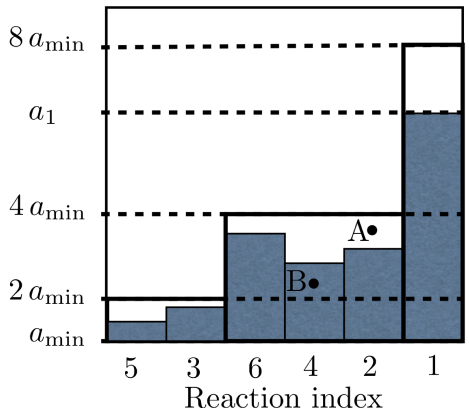



Figure 2.3: Illustration of the binning of the reaction propensities in SSA-CR. In this example the reaction network has six reactions and hence six propensities. The heights of the shaded rectangular bars indicate the magnitudes of the propensities of the reactions with the corresponding index. Points A and B are the examples used in the main text to illustrate rejection sampling.

The computational cost of SSA-CR is $O(G_{\text{a}})$ for weakly coupled reaction networks. This is due to step 2.1 in Table 2.6 that involves a linear search over the $G_{\text{a}}$ bins to identify the bin containing the index of the next reaction. In cases where the ratio of maximum to minimum non-zero propensity is constant, rendering $G_{\text{a}}$ $O(1)$, the computational cost of SSA-CR reduces to $O(1)$. For strongly coupled reaction networks, the computational cost of SSA-CR is $O(M)$ since the degree of coupling of a strongly coupled network is $O(M)$ and hence the cost of the update step (Step 5 in Table 2.6) because $O(M)$ dominates the overall cost.

1. Set $t \leftarrow 0$; initialize $\boldsymbol{n}$, $a_\mu \forall \mu$, $a$, and the dependency graph. Sort the $a_\mu$'s into $G_\mathrm{a}$ bins as described in Sec. 2.3.7. Initialize $a_\mathrm{min}$, $a_\mathrm{max}$ and set $\Delta a \leftarrow 0$
2. Sample $\mu$ using composition-rejection sampling according to Eq. 2.26:
   2.1. Composition step: Generate a uniform random number $r_1 \in [0, 1)$ and perform a linear search over the $G_\mathrm{a}$ bins to sample the bin index $b$ containing the next reaction according to Eq. 2.29
   2.2. Rejection step: Generate a uniform random number $r_2 \in [0,\, 2^b a_\mathrm{min})$ and a uniform random integer between 1 and the number of elements in bin $b$. If the $r_3^\mathrm{th}$ element in bin $b$ is greater than or equal to $r_2$, the corresponding reaction index is chosen as the index of the next reaction $\mu$. If the inequality is not satisfied, repeat the rejection step
3. Sample $\tau$ according to Eq. 2.25: generate a uniform random number $r_2 \in [0, 1)$ and compute $\tau \leftarrow -a^{-1} \log(r_2)$
4. Update: $\boldsymbol{n} \leftarrow \boldsymbol{n} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction $\mu$
5. For all reactions $i$ whose propensities change upon firing of reaction $\mu$ (i.e., the destination nodes of all directed edges leaving node $\mu$ in the dependency graph):
   5.1. Update $\Delta a \leftarrow \Delta a - a_i$
   5.2. Recompute the propensity $a_i$ and reassign the bin membership of $a_i$ if needed
   5.3. Update $\Delta a \leftarrow \Delta a + a_i$
6. Set $a \leftarrow a + \Delta a$
7. $t \leftarrow t + \tau$ and set $\Delta a \leftarrow 0$; go to step 2

Table 2.6: Algorithm for the SSA with composition-rejection sampling (SSA-CR)

### 2.3.8 Summary of the computational costs and memory requirements of these exact SSA formulations

**Computational cost:** The computational costs of all the aforementioned exact SSA formulations are $O(M)$ for strongly coupled reaction networks [28]. For weakly coupled networks, however, some are significantly more efficient and can be $O(\log_2 M)$ or even $O(1)$ [28, 1, 18].

**Memory requirement:** The memory requirements of all aforementioned exact SSA formulations are $O(M)$ when not using a dependency graph. When using a dependency graph, this can be $O(M^2)$ for strongly coupled networks, but is always $O(M)$ for weakly coupled ones.

## 2.4 Partial-propensity exact SSA

The computational cost of exact SSA formulations depends on the property of the chemical reaction network. To predict the computational cost of exact SSA formulations, we classify reaction networks into two classes: *strongly coupled* and *weakly coupled*. Any chemical reaction network with $N$ species and $M$ reactions can be represented by its dependency graph. Each node in this graph represents a chemical reaction and a directed edge is drawn from node $p$ to node $q$ if firing of reaction $p$ affects the copy number of any of the reactants of reaction $q$. In this representation, we quantify the degree of coupling of the reaction network as the maximum number of edges leaving any node, i.e., the maximum out-degree of the dependency graph. We define weakly coupled networks as those in which the degree of coupling is bounded by a constant with increasing network size. Strongly coupled networks have a degree of coupling that increases unboundedly with network size. The computational cost of exact SSA formulations depends on the coupling class of the reaction network. For weakly coupled reaction networks the computational cost (CPU time) has been reduced to $O(1)$ [18] under the assumption that the ratio of maximum to minimum propensity is bounded by a constant. For strongly coupled networks, however, the computational cost of exact SSAs remains $O(M)$. For details see Chapter **??**.

We present here a new class of exact SSA formulations using the novel concept of partial propensities. In doing so, we limit ourselves to networks of elementary reactions under the premise that non-elementary reactions can be broken down to elementary ones at the expense of an increase in network size [3, 31, 32] (see Appendix **??**). In Sec. 2.4.1 we introduce partial propensities. In Sec. 2.4.2 we introduce the concept of partial-propensity formulations. Using partial propensities we first describe the partial-propensity direct method (PDM) in Sec. 2.4.3. The computational cost of PDM is $O(N)$ (i.e., linear in the number of chemical species) irrespective of the degree of coupling of the reaction network. In Sec. 2.4.5 we present the partial-propensity SSA with composition-rejection sampling (PSSA-CR) that further reduces the computational cost for weakly-coupled networks to $O(1)$. Finally, we summarize the partial-propensity formulations as a family of SSAs with algorithmic building blocks that naturally constitute the different formulations.

### 2.4.1 The partial propensity of a reaction

We define the partial propensity of a reaction with respect to one of its reactants as the propensity per molecule of this reactant. For example, the partial propensity $\pi_\mu^{(i)}$ of reaction $\mu$ with respect to (perhaps the only) reactant $\mathrm{S}_i$ is $a_\mu/n_i$, where $a_\mu$ is the propensity of reaction $\mu$ and $n_i$ is the number of molecules of $\mathrm{S}_i$. The partial propensities of the three elementary reaction types are:

- Bimolecular reactions ($\mathrm{S}_i + \mathrm{S}_j \to$ Products): $a_\mu = n_i\, n_j\, c_\mu$ and $\pi_\mu^{(i)} = n_j\, c_\mu$, $\pi_\mu^{(j)} = n_i\, c_\mu$. If both reactants are of the same species, i.e. $\mathrm{S}_i = \mathrm{S}_j$, only one partial propensity exists, $\pi_\mu^{(i)} = \frac{1}{2}(n_i - 1)c_\mu$ because the reaction degeneracy is $\frac{1}{2}n_i(n_i - 1)$.

- Unimolecular reactions ($\mathrm{S}_i \to$ Products): $a_\mu = n_i\, c_\mu$ and $\pi_\mu^{(i)} = c_\mu$.

- Source reactions ($\emptyset \to$ Products): $a_\mu = c_\mu$ and $\pi_\mu^{(0)} = c_\mu$.

### 2.4.2 The concept of partial-propensity SSA formulations

Partial-propensity formulations use partial propensities and group them in order to sample the index of the next reaction and to update the partial propensities after a reaction has fired. For the sampling step, the partial propensities are grouped according to the index of the factored-out reactant, yielding at most $N+1$ groups of size $O(N)$. Sampling then proceeds in two steps: we first sample the index of the group before sampling the actual partial propensity inside that group. This sampling procedure can also be interpreted as sampling reaction partners. The first step involves sampling a reactant before sampling its reaction partner. This grouping scheme reduces the number of operations needed for sampling the next reaction using a concept that is reminiscent of cell lists [33].

After the selected reaction has been executed, we use a dependency graph over species, rather than reactions, to find all partial propensities that need to be updated. The dependency graph over species acts as a pointer to the partial propensities that need to be updated upon change in population of a certain species. This is possible because partial propensities depend on the population of at most one species, and it is analogous to a Verlet list [34]. This limits the number of updates to $O(N)$. In addition, partial propensities of unimolecular reactions are constant and never need to be updated. In weakly coupled networks, where the degree of coupling is $O(1)$, the scaling of the computational cost of the update becomes equal to that of methods that use dependency graphs over reactions, such as SSA-CR [18] (see Sec. 2.3.7), ODM [15] (see Sec. 2.3.4), and SDM [16] (see Sec. 2.3.5).

We illustrate the concept of partial propensity methods using a simple protein aggregation example. Consider proteins that aggregate to form at most tetrameric complexes. There are $N = 4$ species in the reaction network: monomers ($S_1$), dimers ($S_2$), trimers ($S_3$), and tetramers ($S_4$). All species except tetramers can aggregate in all possible combinations to form multimeric complexes (4 bimolecular reactions). In addition, all multimeric complexes can dissociate into any possible combination of two smaller units (4 unimolecular reactions) and monomers are constantly produced (1 source reaction). The reaction network is given by the following 9 reactions

$$\begin{array}{lrcl}
\text{Reaction 1}: & \emptyset & \rightarrow & S_1 \\
\text{Reaction 2}: & S_1 + S_1 & \rightarrow & S_2 \\
\text{Reaction 3}: & S_1 + S_2 & \rightarrow & S_3 \\
\text{Reaction 4}: & S_1 + S_3 & \rightarrow & S_4 \\
\text{Reaction 5}: & S_2 + S_2 & \rightarrow & S_4 \\
\text{Reaction 6}: & S_2 & \rightarrow & S_1 + S_1 \\
\text{Reaction 7}: & S_3 & \rightarrow & S_1 + S_2 \\
\text{Reaction 8}: & S_4 & \rightarrow & S_1 + S_3 \\
\text{Reaction 9}: & S_4 & \rightarrow & S_2 + S_2.
\end{array} \tag{2.30}$$

This reaction network is described by $M = 9$ partial propensities ($\pi_1^{(0)}$), ($\pi_2^{(1)}$, $\pi_3^{(1)}$, $\pi_4^{(1)}$), ($\pi_5^{(2)}$, $\pi_6^{(2)}$), ($\pi_7^{(3)}$), ($\pi_8^{(4)}$, $\pi_9^{(4)}$). Grouping the partial propensities according to the index of the factored-out reactant given in the superscript, we obtain 5 ($= N + 1$) groups as indicated by the parentheses. Along with each group, we store the sum of all partial propensities inside it. We first sample the group that contains the next reaction before finding the corresponding partial propensity inside that group. Assume that in our example reaction 7 is to fire next. Different search algorithms can be used for this task. Using linear search, for instance, the search depth to find the group index is 4 and the search depth to find the partial propensity ($\pi_7^{(3)}$) is 1. Linear search thus requires 5

operations to sample the next reaction in this network of 9 reactions. The average linear search depth for sampling the next reaction in this example is $37/9 \approx 4.1$, if all reaction propensities are equal.

### 2.4.3 The partial-propensity direct method (PDM)

In PDM, the index of the next reaction $\mu$ is sampled using linear search over groups and subsequently over elements in the group. The sampling procedure is algebraically equivalent to that of Gillespie's direct method (DM). The novelties in PDM are the use of partial propensities and efficient data structures that reduce the number of operations needed to sample $\mu$ and to update the partial propensities. The time to the next reaction $\tau$ is sampled as in DM.

#### 2.4.3.1 Detailed description

All partial propensities are stored in the "partial-propensity structure" $\mathbf{\Pi} = \{\mathbf{\Pi}_i\}_{i=0}^{N}$ as a one-dimensional array of one-dimensional arrays $\mathbf{\Pi}_i$. Each array $\mathbf{\Pi}_i$ contains the partial propensities belonging to group $i$. The partial propensities of source reactions are stored as consecutive entries of the $0^{\text{th}}$ array $\mathbf{\Pi}_0$. The partial propensities of all reactions that have species $\mathrm{S}_1$ as one of its reactants are stored as consecutive entries of $\mathbf{\Pi}_1$. In general, the $i^{\text{th}}$ array $\mathbf{\Pi}_i$ contains the partial propensities of all reactions that have $\mathrm{S}_i$ as a reactant, provided these reactions have not yet been included in any of the previous $\mathbf{\Pi}_{j<i}$. That is, out of the two partial propensities of a reaction $\mu$ with $\mathrm{S}_i$ and $\mathrm{S}_j$ as reactants, $\pi_\mu^{(i)}$ is part of $\mathbf{\Pi}_i$ if $i < j$, and $\pi_\mu^{(j)}$ is not stored anywhere. In order for $\mathbf{\Pi}$ to be independent of the numbering of the reactants, we first renumber the species such that $\mathrm{S}_i$ is the species involved as a reactant in $i^{\text{th}}$-most reactions. This ordering also minimizes the number of required update operations. $\mathbf{\Pi}$ needs to be constructed only once, at the beginning of a simulation. The steps to automatically build $\mathbf{\Pi}$ from the stoichiometry matrix are outlined in Table 2.7.

Since the different $\mathbf{\Pi}_i$'s can be of different length, storing them as an array of arrays is more (memory) efficient than using a matrix (i.e., a two-dimensional array). The reaction indices of the partial propensities in $\mathbf{\Pi}$ are stored in a look-up table $\mathbf{L} = \{\mathbf{L}_i\}_{i=0}^{N}$, which is also an array of arrays. This makes every reaction $\mu$ identifiable by a unique pair of indices, a group index $I$ and an element index $J$, such that the partial propensity of reaction $\mu = \mathrm{L}_{I,J}$ is stored in $\Pi_{I,J}$.

We further define the "group-sum array" $\mathbf{\Lambda}$, storing the sums of the partial propensities in each group $\mathbf{\Pi}_i$, thus $\Lambda_i = \sum_j \Pi_{i,j}$, $i = 0, \ldots, N$. In addition, we also define $\mathbf{\Sigma}$, the array of the total propensities of all groups, as $\Sigma_i = n_i \Lambda_i$, $i = 0, \ldots, N$, and set the population $n_0$ of the reservoir in the source reactions to 1. The total propensity of all reactions is then $a = \sum_{i=0}^{N} \Sigma_i$. The use of $\mathbf{\Lambda}$ avoids having to recompute the sum of all partial propensities in $\mathbf{\Pi}_i$ after one of them has changed. Rather, the same change is also applied to $\Lambda_i$, and computing the new $\Sigma_i$ only requires a single multiplication by $n_i$. Using these data structures and a single uniformly distributed random number $r_1 \in [0, 1)$, the next reaction $\mu$ can efficiently be sampled in two steps: (1) sampling the group index $I$ such that

$$I = \min \left[ I' \ : \ r_1 a < \sum_{i=0}^{I'} \Sigma_i \right] \tag{2.31}$$

1. Initialize the reactant stoichiometry matrix $\boldsymbol{\nu}^-$, the initial population $\mathbf{n}(0)$, and the specific probability rates $\mathbf{c}$. Reorder all stoichiometry matrices such that the $i^{\text{th}}$ row corresponds to the species involved as a reactant in $i^{\text{th}}$-most reactions.
2. Using $\boldsymbol{\nu}^-$, build a list of all reactants in each reaction. For reaction $\mu$, the reactants have a non-zero entry in $\boldsymbol{\nu}^-_\mu$. If all species have a zero reactant stoichiometry then the reactant index is 0 and the reaction is a source reaction.
3. For each reaction, go through the list of reactants:
   - 3.1 If the number of distinct reactants in a reaction is 2, compute the partial propensity of this reaction by factoring out the population of the species with the *smaller* index $i$ from the full reaction propensity. Append this partial propensity to $\boldsymbol{\Pi}_i$.
   - 3.2 If the number of reactants in a reaction is 1, then check
     - 3.2.1 If it is a biomolecular reaction between the same species $\mathrm{S}_i$, store the corresponding partial propensity in $\boldsymbol{\Pi}_i$.
     - 3.2.2 If it is a unimolecular reaction with only species $\mathrm{S}_i$ as a reactant, store the partial propensity in $\boldsymbol{\Pi}_i$.
     - 3.2.3 If it is a source reaction ($i = 0$), store the partial propensity in $\boldsymbol{\Pi}_0$.
4. Stop.

Table 2.7: Algorithm for constructing the partial-propensity structure $\boldsymbol{\Pi}$

and (2) sampling the element index $J$ in $\boldsymbol{\Pi}_I$ such that

$$J = \min\left[J' \;:\; r_1 a < \sum_{j=1}^{J'} n_I \Pi_{I,j} + \left(\sum_{i=0}^{I} \Sigma_i\right) - \Sigma_I\right] . \tag{2.32}$$

(see Appendix **??** for a proof of the equivalence of this sampling scheme to that of DM.) Using the temporary variables

$$\Phi = \sum_{i=0}^{I} \Sigma_i, \qquad \Psi = \frac{r_1 a - \Phi + \Sigma_I}{n_I} , \tag{2.33}$$

Eq. 2.32 can be efficiently implemented as

$$J = \min\left[J' \;:\; \Psi < \sum_{j=1}^{J'} \Pi_{I,j}\right] . \tag{2.34}$$

The indices $I$ and $J$ are then translated back to the reaction index $\mu$ using the look-up table $\mathbf{L}$, thus $\mu = \mathrm{L}_{I,J}$.
To execute a sampled reaction, $\boldsymbol{n}$, $\boldsymbol{\Pi}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\Sigma}$ need to be updated. This is efficiently done using three update structures:

$\mathbf{U}^{(1)}$ is an array of $M$ arrays, where the $i^{\text{th}}$ array contains the indices of all species involved in the $i^{\text{th}}$ reaction.

$\mathbf{U}^{(2)}$ is an array of $M$ arrays containing the corresponding stoichiometry (the change in population of each species upon reaction) of the species stored in $\mathbf{U}^{(1)}$. $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ constitute the sparse representation of the stoichiometry matrix $\boldsymbol{\nu}$.

$\mathbf{U}^{(3)}$ is an array of $N$ arrays, where the $i^{\text{th}}$ array contains the indices of all entries in $\boldsymbol{\Pi}$ that depend on $n_i$, thus:

$$\mathbf{U}^{(3)} = \begin{cases} \mathbf{U}_1^{(3)} = \begin{bmatrix} \left(i_1^1, j_1^1\right) & \left(i_2^1, j_2^1\right) & \dots & \dots & \dots \end{bmatrix} \\ \mathbf{U}_2^{(3)} = \begin{bmatrix} \left(i_1^2, j_1^2\right) & \left(i_2^2, j_2^2\right) & \dots \end{bmatrix} \\ \vdots \\ \mathbf{U}_N^{(3)} = \begin{bmatrix} i_1^N, j_1^N & i_2^N, j_2^N & \dots & \dots \end{bmatrix}. \end{cases} \tag{2.35}$$

When a reaction is executed, the populations of the species involved in this reaction change. Hence, all entries in $\boldsymbol{\Pi}$ that depend on these populations need to be updated. After each reaction, we use $\mathbf{U}^{(1)}$ to determine the indices of all species involved in this reaction. The stoichiometry is then looked up in $\mathbf{U}^{(2)}$ and the population $\boldsymbol{n}$ is updated. Subsequently, $\mathbf{U}^{(3)}$ is used to locate the affected entries in $\boldsymbol{\Pi}$ and recompute them. The two data structures $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ are a sparse representation of the stoichiometry matrix, and $\mathbf{U}^{(3)}$ represents the dependency graph over species. Since the partial propensities of unimolecular and source reactions are constant and need never be updated, $\mathbf{U}^{(3)}$ only contains the indices of the partial propensities of bimolecular reactions. The size of $\mathbf{U}^{(3)}$ is a factor of $O(N)$ smaller than that of the corresponding dependency graph over reactions, since partial propensities depend on the population of at most one species. Figure 2.4 summarizes the data structures used in PDM for an example reaction network. The complete algorithm is given in Table 2.8. Overall, PDM's computational cost is $O(N)$ and its memory requirement is $O(M)$, irrespective of the degree of coupling of the simulated network (see Sec. 2.4.4.1).

1. Initialization: set $t \leftarrow 0$; initialize $\boldsymbol{n}$, $\boldsymbol{\Pi}$, $\boldsymbol{\Lambda}$, $\boldsymbol{\Sigma}$; $a \leftarrow \sum_{i=0}^{N} \Sigma_i$; $\Delta a \leftarrow 0$; generate $\mathbf{L}$, $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$
2. Sample $\mu$: generate a uniform random number $r_1 \in [0,1)$ and determine the group index $I$ and the element index $J$ according to Eqs. 2.31, 2.33, and 2.34; $\mu \leftarrow \mathrm{L}_{I,J}$
3. Sample $\tau$: generate a uniform random number $r_2 \in [0,1)$ and compute the time to next reaction $\tau$ as $\tau \leftarrow a^{-1} \ln(r_2^{-1})$
4. Update $\boldsymbol{n}$: for each index $k$ of $\mathbf{U}_\mu^{(1)}$, $l \leftarrow \mathrm{U}_{\mu,k}^{(1)}$ and $n_l \leftarrow n_l + \mathrm{U}_{\mu,k}^{(2)}$
5. Update $\boldsymbol{\Pi}$, $\boldsymbol{\Lambda}$, $\boldsymbol{\Sigma}$ and compute $\Delta a$, the change in $a$:

   For each index $k$ of $\mathbf{U}_\mu^{(1)}$, do:

   5.1. $l \leftarrow \mathrm{U}_{\mu,k}^{(1)}$

   5.2. For each index $m$ of $\mathbf{U}_l^{(3)}$, do:

   5.2.1. $(i_m^l,\, j_m^l) \leftarrow \mathrm{U}_{l,m}^{(3)}$ (Eq. 2.35)

   5.2.2. $\Pi_{i_m^l, j_m^l} \leftarrow \Pi_{i_m^l, j_m^l} + c_{\mu'} \mathrm{U}_{\mu,k}^{(2)},\ \mu' = \mathrm{L}_{i_m^l,\, j_m^l}$ if $l \neq i_m^l$
   $\Pi_{i_m^l, j_m^l} \leftarrow \Pi_{i_m^l, j_m^l} + \frac{1}{2} c_{\mu'} \mathrm{U}_{\mu,k}^{(2)},\ \mu' = \mathrm{L}_{i_m^l,\, j_m^l}$ if $l = i_m^l$

   5.2.3. $\Lambda_{i_m^l} \leftarrow \Lambda_{i_m^l} + c_{\mu'} \mathrm{U}_{\mu,k}^{(2)},\ \mu' = \mathrm{L}_{i_m^l,\, j_m^l}$ if $l \neq i_m^l$
   $\Lambda_{i_m^l} \leftarrow \Lambda_{i_m^l} + \frac{1}{2} c_{\mu'} \mathrm{U}_{\mu,k}^{(2)},\ \mu' = \mathrm{L}_{i_m^l,\, j_m^l}$ if $l = i_m^l$

   5.2.4. $\Sigma_{\mathrm{temp}} \leftarrow \Sigma_{i_m^l}$

   5.2.5. $\Sigma_{i_m^l} \leftarrow n_{i_m^l} \Lambda_{i_m^l}$

   5.2.6. $\Delta a \leftarrow \Delta a + \Sigma_{i_m^l} - \Sigma_{\mathrm{temp}}$

   5.3. $\Delta a \leftarrow \Delta a + n_l \Lambda_l - \Sigma_l;\ \Sigma_l \leftarrow n_l \Lambda_l$
6. Update $a$ and increment time: $a \leftarrow a + \Delta a;\ \Delta a \leftarrow 0;\ t \leftarrow t + \tau$
7. Go to step 2

Table 2.8: Detailed algorithm for the partial-propensity direct method PDM.

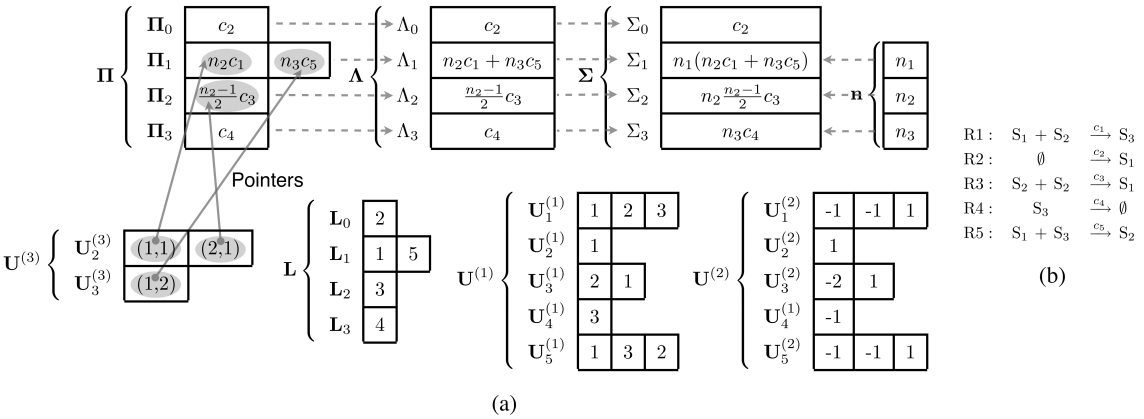



Figure 2.4: (a) Illustration of the data structures in PDM for the example reaction network shown in (b). Note that there may be arrays $\mathbf{\Pi}_i$, $i = 1, \ldots, N$, containing at most one negative entry if the corresponding $n_i = 0$. Indeed, in this example, $\Pi_{2,1} < 0$ and $\Lambda_2 < 0$ if $n_2 = 0$. This, however, poses no problem in sampling $I$ and $J$ as all $\Sigma_i$ for which $n_i = 0$ are zero and hence the corresponding group indices $I$ are never selected.

### 2.4.4 The sorting partial-propensity direct method (SPDM)

The sorting partial-propensity direct method (SPDM) is the partial-propensity variant of SDM [16]. In SPDM, the group and element indices $I$ and $J$ are bubbled up whenever the reaction $\mu = \mathrm{L}_{I,J}$ fires. The reordered indices are stored in an array for $I$, and an array of arrays of the size of $\mathbf{\Pi}$ for the $J$'s. This requires an additional $N + M$ memory, but further reduces the average search depth to sample the next reaction, especially for multiscale (stiff) networks. The computational cost of SPDM is also $O(N)$, but with a possibly reduced pre-factor (see Sec. 2.4.4.1).

#### 2.4.4.1 Computational cost and memory requirements

The computational cost of PDM is governed by the following steps: (a) sampling the index of the next reaction and (b) updating the population $\boldsymbol{n}$ and the partial-propensity structure $\mathbf{\Pi}$. The scaling of the computational cost of SPDM is the same as that of PDM. In terms of absolute runtimes SPDM, however, is expected to be more efficient than PDM especially for multiscale reaction networks.

**Computational cost of sampling the index of the next reaction:** For any chemical reaction network with $N$ species, the number of arrays in the partial-propensity structure $\mathbf{\Pi}$ is at most $N+1$, which is also the maximum length of $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$. The number of entries in each array $\mathbf{\Pi}_i$ is $O(N)$, since any species can react with at most $N$ species in bimolecular reactions and undergo at most $O(N)$ unimolecular reactions. Sampling the index of the next reaction involves two steps: (a) a linear search for the group index $I$ in $\mathbf{\Sigma}$ and (b) a linear search for the element index $J$ in $\mathbf{\Pi}_I$. Since $\mathbf{\Sigma}$ is at most of length $N + 1$, the first step is $O(N)$. The second step is also $O(N)$, since the number of elements in $\mathbf{\Pi}_i$ is $O(N)$. The overall computational cost of sampling the next reaction is thus $O(N)$ for networks of any degree of coupling.

**Computational cost of the update:** Let the maximum number of chemical species involved in any reaction (as reactants or products) be given by the constant $s$. Let us assume that $s$ is $O(1)$ with increasing system size. This assumption is not restrictive since the number of species involved in a reaction is unlikely to increase with system size. The computational cost of updating $\boldsymbol{n}$ is thus $s \in O(1)$. In PDM, only the partial propensities of bimolecular reactions need to be updated. The total number of entries in the third update structure $\mathbf{U}^{(3)}$ is, thus, equal to the number of bimolecular reactions. In addition, the total number of entries in $\mathbf{\Pi}$ that depend on any $n_i$ is always less than or equal to $N$, as any species $\mathrm{S}_i$ can only react with itself and the remaining $N-1$ species in bimolecular reactions. Therefore, the upper bound for the total number of partial propensities in $\mathbf{\Pi}$ to be updated after executing any reaction is $sN \in O(N)$.
In summary, the computational cost of PDM is $O(N)$, irrespective of the degree of coupling of the reaction network (see Sec. 2.4.4.2 for benchmark results).
The memory requirement of PDM is given by the total size of the data structures $\boldsymbol{n}$, $\mathbf{\Pi}$, $\mathbf{L}$, $\mathbf{\Lambda}$, $\mathbf{\Sigma}$, $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$.
The partial-propensity structure $\mathbf{\Pi}$ and the look-up table $\mathbf{L}$ have the same size. Since every reaction is accounted for exactly once, each structure requires $O(M)$ memory. $\mathbf{\Lambda}$, $\boldsymbol{n}$, and $\mathbf{\Sigma}$ are all at most of length $N+1$ and thus require $O(N)$ memory. The sizes of $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ are $O(M)$, and the size of $\mathbf{U}^{(3)}$ is proportional the number of bimolecular reactions and, hence, $O(M)$ if all reactions are bimolecular.
In summary, the memory requirement of PDM is $O(M)$. SPDM requires an additional $N+M$ memory to store the reordered index lists.

#### 2.4.4.2 Benchmarks

We benchmark the computational performance of PDM and SPDM using four chemical reaction networks that are prototypical of: (a) strongly coupled reaction networks, (b) strongly coupled reaction networks comprising only bimolecular reactions, (c) weakly coupled reaction networks, and (d) multiscale biological networks. The first two benchmarks consider strongly coupled networks where the degree of coupling $d_{\mathrm{c}}$ scales with system size (see column "Maximum" under "Degree of coupling of nodes ($d_\mu$)" in Table 2.9). The first benchmark consists of a colloidal aggregation model. The second benchmark considers a network of only bimolecular reactions, where none of the partial propensities are constant. In the third benchmark, we compare PDM and SPDM to SDM on the linear chain model, a weakly coupled reaction network with the minimal degree of coupling, for which SDM was reported to be very efficient [15, 16]. The fourth benchmark considers the heat-shock response model, a small multiscale (stiff) biological reaction network of fixed size.
All tested SSA formulations are implemented in C++ using the random-number generator of the GSL library and compiled using the GNU C++ compiler version 4.0.1 with the O3 optimization flag. All timings are determined using a nanosecond-resolution timer (the `mach_absolute_time()` system call) on a MacOS X 10.4.11 workstation with a 3 GHz dual-core Intel Xeon processor, 8 GB of memory, and a 4 MB L2 cache. For each test case we report both the memory requirement and the average CPU time per reaction (i.e., per time step), $\Theta$. $\Theta$ is defined as the CPU time (identical to wall-clock time in our case) needed to simulate the system up to final time $T$, divided by the total number of reactions executed during the simulation, and averaged over independent runs. The time $\Theta$ does not include the initialization of the data structures (step 1 in Table 2.8) as this is done only once and is not part of the time loop.
We explain the benchmark results in terms of the computational cost of the individual steps of the

| Model | Number of species ($N$) | Number of reactions ($M$) | Degree of coupling of nodes ($d_\mu$) | | |
|---|---|---|---|---|---|
| | | | Minimum ($\min\{d_\mu\}$) | Average $\left(\frac{\sum_{\mu=1}^{M} d_\mu}{M}\right)$ | Maximum ($d_c = \max\{d_\mu\}$) |
| CA | $N$ | $\left\lfloor \frac{N^2}{2} \right\rfloor$ | $\frac{4N-1}{3}$ | $2.3N - 4.7$ | $3N - 7$ |
| NB | $N$ | $\frac{N(N-1)}{2}$ | $4N - 10$ | $4N - 10$ | $4N - 10$ |
| LC | $N$ | $N - 1$ | $1^{(*)}$ | $2 - \frac{1}{N-1} \approx 2$ | 2 |
| HSR | 28 | 61 | 1 | 5.9 | 11 |

Table 2.9: Properties of the benchmark cases. The number of species, number of reactions, and minimum, average, maximum out-degree (degree of coupling) of the dependency graph are given for the benchmark cases defined in Sections 2.4.4.3, 2.4.4.4, 2.4.4.5 and 2.4.4.6: the colloidal aggregation model (CA), the network of bimolecular reactions (NB), the linear chain model (LC), and the heat-shock response model (HSR). ($^*$) In the linear chain model the degree of coupling is 1 only for the last reaction, since its product is not a reactant anywhere else.

algorithms. We distinguish three steps: (a) sampling the index of the next reaction, (b) updating the population, and (c) updating the partial propensities (for PDM and SPDM) or the propensities (for SDM). The computational costs of these steps are quantified separately and the overall timings are then explained as a weighted sum of:

- $\mathcal{C}_\mu$: The number of operations required to sample the index of the next reaction (for PDM, this is step 2 in Table 2.8).
- $\mathcal{C}_{\mathbf{n}}$: The number of elements of the population $\boldsymbol{n}$ that need to be updated after executing a reaction (for PDM, this is step 4 in Table 2.8).
- $\mathcal{C}_{\mathrm{P}}$: The number of (partial) propensities that need to be updated after executing a reaction (for PDM, this is step 5.2.2 in Table 2.8).

The expressions for these elementary costs are given in Table 2.10 as determined by independently fitting models for the scaling of the algorithms to the measured operation counts, averaged over 100 independent runs of each test problem. In all cases, the models used for the computational cost explain the data with a correlation coefficient of at least 0.98. The benchmark results are then explained by fitting the weights of the cost superposition $a\mathcal{C}_\mu + b\mathcal{C}_{\mathbf{n}} + c\mathcal{C}_{\mathrm{P}}$ to the measured scaling curves $\Theta(N)$ using the expressions given in Table 2.10. In order to preserve the relative weights of the data points, all fits are done on a linear scale, even though the results are plotted on a logarithmic scale for two of the benchmarks. All these fits also have a correlation coefficient of at least 0.98. Explaining the timing results as a superposition of elementary costs allows determining which part of an algorithm is responsible for a particular speedup or scaling behavior, and what the relative contributions of the three algorithmic steps are to the overall computational cost.
The memory requirements of the algorithms are reported in Table 2.11 for all benchmark cases. These numbers were derived analytically from the size of the individual data structures.

| | PDM | | | SPDM | | |
|---|---|---|---|---|---|---|
| | $\mathcal{C}_\mu$ | $\mathcal{C}_\mathbf{n}$ | $\mathcal{C}_\mathrm{P}$ | $\mathcal{C}_\mu$ | $\mathcal{C}_\mathbf{n}$ | $\mathcal{C}_\mathrm{P}$ |
| CA | $0.49N + 2.0$ | 3 | $5.2N^{0.5} - 8.1$ | $0.45N + 0.38$ | 3 | $5.2N^{0.5} - 8.1$ |
| NB | $0.97N - 1.3$ | 4 | $1.6N - 3.2$ | $0.94N - 4.7$ | 4 | $1.6N - 3.2$ |
| LC | $0.50N + 1.0$ | 2 | 0 | $1.0N^{0.5} + 0.79$ | 2 | 0 |
| HSR | 13 | 3 | 2.2 | 3.7 | 3 | 2.2 |

| | SDM | | |
|---|---|---|---|
| | $\mathcal{C}_\mu$ | $\mathcal{C}_\mathbf{n}$ | $\mathcal{C}_\mathrm{P}$ |
| CA | $0.14N^2 + 1.2N - 9.9$ | $N$ | $2.8N - 10$ |
| NB | $0.33N^2 - 0.044N + 0.51$ | $N$ | $4.0N - 10$ |
| LC | $1.0N^{0.5} - 0.21$ | $N$ | 2 |
| HSR | 2.9 | 28 | 8.2 |

Table 2.10: Number of compute operations needed by the different algorithms (PDM, SPDM, SDM) for the different test cases (CA: colloidal aggregation model; NB: network of bimolecular reactions; LC: linear chain model; HSR: heat-shock response model). $\mathcal{C}_\mu$ is the average number of operations needed to sample the next reaction $\mu$. $\mathcal{C}_\mathbf{n}$ is the average number of entries in the population $\boldsymbol{n}$ that need to be updated after any reaction. $\mathcal{C}_\mathrm{P}$ is the average number of partial propensities (or propensities for SDM) that need to be updated after any reaction. The operation counts are averaged over all reactions executed during 100 independent runs of each benchmark over the range of $N$ shown in Fig. 2.5. The average numbers are then fitted with the models given here (with correlation coefficient of at least 0.98 in all cases). See Fig. 2.6 for the distribution of the number of updates.

| | PDM/SPDM | | | | |
|---|---|---|---|---|---|
| | $\mathbf{n}$, $\mathbf{\Lambda}$, $\mathbf{\Sigma}$ | $\mathbf{\Pi}$, $\mathbf{L}$, $\mathbf{c}$ | $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$ | $\mathbf{U}^{(3)}$ | Total |
| CA | $N$ | $\left\lfloor \frac{N^2}{2} \right\rfloor$ | $3\left\lfloor \frac{N^2}{2} \right\rfloor$ | $2\left\lfloor \frac{N^2}{4} \right\rfloor$ | $O(N^2) = O(M)$ |
| NB | $N$ | $\frac{N(N-1)}{2}$ | $4\frac{N(N-1)}{2}$ | $2\frac{N(N-1)}{2}$ | $O(N^2) = O(M)$ |
| LC | $N$ | $N-1$ | $2(N-1)$ | 0 | $O(N) = O(M)$ |
| HSR | 28 | 61 | 133 | 24 | 557 |

| | SDM | | | | |
|---|---|---|---|---|---|
| | $\mathbf{n}$ | $\mathbf{c}$, $\mathbf{a}$ | dependency graph | $\boldsymbol{\nu}$ | Total |
| CA | $N$ | $\left\lfloor \frac{N^2}{2} \right\rfloor$ | $1.2N^3 - 2.5N^2 + 2.3N$ | $N\left\lfloor \frac{N^2}{2} \right\rfloor$ | $O(N^3) = O(NM)$ |
| NB | $N$ | $\frac{N(N-1)}{2}$ | $2N^3 - 7N^2 + 5N$ | $\frac{N^2(N-1)}{2}$ | $O(N^3) = O(NM)$ |
| LC | $N$ | $N-1$ | $2(N-1)$ | $N(N-1)$ | $O(N^2) = O(NM)$ |
| HSR | 28 | 61 | 360 | 1708 | 2218 |

Table 2.11: Total amount of computer memory needed by the different algorithms (PDM, SPDM, SDM) for the different test cases (CA: colloidal aggregation model; NB: network of bimolecular reactions; LC: linear chain model; HSR: heat-shock response model). The sizes of all major data structures ($\mathbf{c}$ and $\mathbf{a}$ are the arrays of specific probability rates and reaction propensities, respectively; $\boldsymbol{\nu}$ is the stoichiometry matrix; see Sec. 2.4.3 for other definitions) as well as the total memory requirements are given as determined analytically for all benchmark simulations. SPDM and SDM need additional memory of size $M + N$ and $M$, respectively, for the reordered index lists. This, however, does not change the overall scaling of the total memory requirements.

#### 2.4.4.3 Strongly coupled reaction network: colloidal aggregation model

We use the colloidal aggregation model [35, 36, 37, 38, 39] as a first example of a strongly coupled reaction network. The reaction network of the colloidal aggregation model is defined by:

$$\begin{aligned} \mathrm{S}_n + \mathrm{S}_m &\xrightarrow{c_{n,m}} \mathrm{S}_{n+m} \qquad && n+m = 2,\dots,N \\ \mathrm{S}_{p+q} &\xrightarrow{\bar{c}_{p,q}} \mathrm{S}_p + \mathrm{S}_q \qquad && p+q = 2,\dots,N. \end{aligned} \tag{2.36}$$

For an even number of species $N$, the partial-propensity structure for this network is:

$$\boldsymbol{\Pi} = \begin{cases} \boldsymbol{\Pi}_0 \;=\; (\emptyset) \\ \boldsymbol{\Pi}_1 \;=\; \begin{pmatrix} c_{1,1}\frac{n_1-1}{2} & c_{1,2}n_2 & c_{1,3}n_3 & \dots & c_{1,\frac{N}{2}}n_{\frac{N}{2}} & \dots & c_{1,N-1}n_{N-1} \end{pmatrix} \\ \boldsymbol{\Pi}_2 \;=\; \begin{pmatrix} \bar{c}_{2,1} & c_{2,2}\frac{n_2-1}{2} & c_{2,3}n_3 & \dots & c_{2,\frac{N}{2}}n_{\frac{N}{2}} & \dots & c_{2,N-2}n_{N-2} \end{pmatrix} \\ \vdots \\ \boldsymbol{\Pi}_{\frac{N}{2}} \;=\; \begin{pmatrix} \bar{c}_{\frac{N}{2},1} & \bar{c}_{\frac{N}{2},2} & \dots & \bar{c}_{\frac{N}{2},\frac{N}{4}} & c_{\frac{N}{2},\frac{N}{2}}n_{\frac{N}{2}} \end{pmatrix} \\ \vdots \\ \boldsymbol{\Pi}_N \;=\; \begin{pmatrix} \bar{c}_{N,1} & \bar{c}_{N,2} & \bar{c}_{N,3} & \dots & \dots & \bar{c}_{\frac{N}{2},\frac{N}{2}} \end{pmatrix}. \end{cases} \tag{2.37}$$

For odd $N$, the structure looks similar. This reaction network can be used to model, e.g., colloidal aggregation of solvated proteins, nano-beads, or viruses. For $N$ chemical species it consists of $M = \left\lfloor \frac{N^2}{2} \right\rfloor$ reactions and the maximum out-degree of the dependency graph, i.e., the degree of coupling $d_\mathrm{c}$ of the network is $3N-7$ and hence scales with system size (see Table 2.9).
The colloidal aggregation model is simulated up to time $T = 100$ with specific probability rates $c_{n,m} = 1$ and $\bar{c}_{p,q} = 1$. At time $t = 0$, $n_i = N\delta_{1,i}$. The scaling of $\Theta$ for PDM, SPDM, and SDM with system size is shown in Fig. 2.5a, averaged over 100 independent runs. $\Theta^{\mathrm{PDM}}$ and $\Theta^{\mathrm{SPDM}}$ are $O(N^{0.5})$ for small $N$ (less than about 100) and $O(N)$ for large $N$. $\Theta^{\mathrm{SDM}}$ is $O(N^2)$. The pre-factor of $\Theta^{\mathrm{SPDM}}$ is similar to that of $\Theta^{\mathrm{PDM}}$, since in this network $\mathcal{C}_\mu$ is not significantly reduced by the dynamic sorting (Table 2.10). The memory requirements of PDM and SPDM are $O(N^2) = O(M)$, that of SDM is $O(N^3) = O(NM)$ (Table 2.11).
In summary, the computational costs of both PDM and SPDM are $O(N)$. This scaling is mediated by all three cost components. The use of partial propensities renders the scaling of the sampling cost $\mathcal{C}_\mu$ $O(N)$ (see Table 2.10). The cost $\mathcal{C}_\mathrm{P}$ for updating the partial propensities is $O(N^{0.5})$ (Table 2.10), since the use of partial propensities allows formulating a dependency graph over species, rather than reactions, and unimolecular reactions have constant partial propensities. This leads to a smaller number of updates needed as shown in Fig. 2.6a.

#### 2.4.4.4 Strongly coupled network of bimolecular reactions

The following hypothetical network of bimolecular reactions:

$$\mathrm{S}_n + \mathrm{S}_m \xrightarrow{c_{n,m}} \mathrm{S}_p + \mathrm{S}_q \qquad \begin{cases} n = 1,\dots,N-1\,; \quad m = n+1,\dots,N; \\ p = \min\left[\{1,\dots,N\}\backslash\{n,m\}\right]\,; \\ q = \min\left[\{1,\dots,N\}\backslash\{n,m,p\}\right]\,. \end{cases} \tag{2.38}$$

consists of $M = \frac{N}{2}(N-1)$ strongly coupled bimolecular reactions, such that none of the partial propensities are constant. The partial-propensity structure for this reaction network is:

$$\mathbf{\Pi} = \begin{cases} \mathbf{\Pi}_0 \;=\; (\emptyset) \\ \mathbf{\Pi}_1 \;=\; (c_{1,2}n_2 \quad c_{1,3}n_3 \quad c_{1,4}n_4 \quad \dots \quad c_{1,N}n_N) \\ \mathbf{\Pi}_2 \;=\; (c_{2,3}n_3 \quad c_{2,4}n_4 \quad c_{2,5}n_5 \quad \dots \quad c_{2,N}n_N) \\ \vdots \\ \mathbf{\Pi}_{N-1} \;=\; (c_{N-1,N}n_N) \\ \mathbf{\Pi}_N \;=\; (\emptyset)\;. \end{cases} \tag{2.39}$$

Both the minimum and the maximum out-degrees of the dependency graph in this case are $4N-10$, scaling faster with $N$ than in the colloidal aggregation model (see Table 2.9).
We simulate this network up to time $T = 0.001$ with all specific probability rates $c_i = 1$. At $t = 0$, $n_i = 100(\delta_{N-4,i} + \delta_{N-3,i} + \delta_{N-2,i} + \delta_{N-1,i} + \delta_{N,i})$. The scaling of $\Theta$ for PDM, SPDM, and SDM with system size is shown in Fig. 2.5b, averaged over 100 independent runs. $\Theta^{\mathrm{PDM}}$ and $\Theta^{\mathrm{SPDM}}$ are $O(N)$, whereas $\Theta^{\mathrm{SDM}}$ is $O(N^2)$. The pre-factors of PDM and SPDM are comparable. The memory requirements of PDM and SPDM are $O(N^2) = O(M)$, that of SDM is $O(N^3) = O(NM)$ (see Table 2.11).
In summary, the computational costs of PDM and SPDM are $O(N)$ for this strongly coupled, purely bimolecular network. The scaling is again mediated by all three cost components. Grouping the partial propensities renders the sampling cost $\mathcal{C}_\mu$ $O(N)$ (see Table 2.10). Because none of the partial propensities are constant, the update costs $\mathcal{C}_\mathrm{P}$ of PDM and SPDM are $O(N)$, as in SDM, albeit with a pre-factor that is $\approx 2.5$ times smaller than that in SDM. One reason for this smaller pre-factor is the smaller number of updates needed upon reactions firing, as shown in Fig. 2.6(b). This is due to the fact that partial propensities of bimolecular reactions depend on the population of only one species, which reduces the number of combinations that need to be updated.

#### 2.4.4.5 Weakly coupled reaction network: linear chain model

We benchmark PDM and SPDM on a weakly coupled model in order to assess their limitations in cases where other SSA formulations might be more efficient. We choose the following linear chain model

$$\mathrm{S}_i \xrightarrow{c_i} \mathrm{S}_{i+1} \qquad i = 1, \dots, N-1\,, \tag{2.40}$$

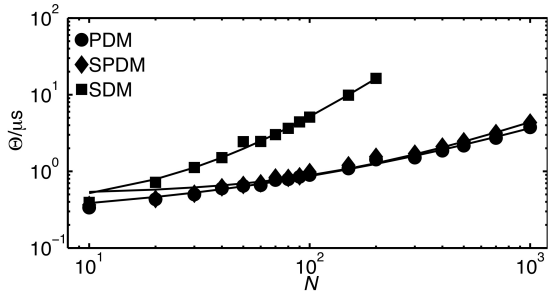
since it is the most weakly coupled reaction network possible, and it has been used as a model for isolated signal transduction networks [40]. For $M$ reactions, it involves the number of species $N = M + 1$, and the maximum out-degree of the dependency graph is constant at the minimum possible value of 2 (see Table 2.9), since every reaction at most influences the population of its only reactant and of the only reactant of the subsequent reaction.
The partial-propensity structure of the linear chain model is given by:

$$\mathbf{\Pi} = \begin{cases} \mathbf{\Pi}_0 \;=\; (\emptyset) \\ \mathbf{\Pi}_1 \;=\; (c_1) \\ \mathbf{\Pi}_2 \;=\; (c_2) \\ \vdots \\ \mathbf{\Pi}_{N-1} \;=\; (c_{N-1}) \\ \mathbf{\Pi}_N \;=\; (\emptyset)\,. \end{cases} \tag{2.41}$$

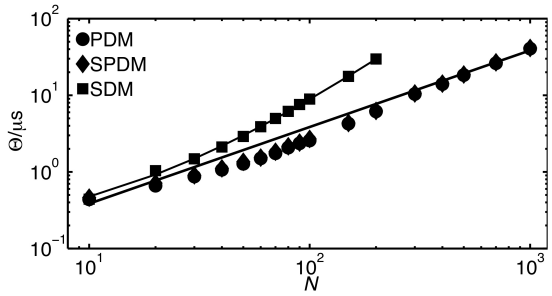
We simulate the linear chain model to a final time of $T = 1000$ with all specific probability rates $c_i = 1$. At time $t = 0$, $n_i = 10000\delta_{1,i}$. Figure 2.5c presents the scaling of the CPU time with system size for PDM, SPDM, and SDM, averaged over 100 independent runs. $\Theta^{\text{PDM}}$ scales linearly with $N$ and $\Theta^{\text{SPDM}}$ with $N^{0.5}$. $\Theta^{\text{SDM}}$ is $O(N)$ with a pre-factor that is more than 4 times larger than that of $\Theta^{\text{PDM}}$. This difference in pre-factor is mainly caused by PDM having smaller $\mathcal{C}_{\mathbf{n}}$ and $\mathcal{C}_{\text{P}}$ (Table 2.10). $\mathcal{C}_\mu$, however, scales worse for PDM than for SDM due to the dynamic sorting in SDM. This is overcome in SPDM, where $\mathcal{C}_\mu$ is $O(N^{0.5})$, as in SDM. The memory requirements of SPDM and PDM are $O(N) = O(M)$, that of SDM is $O(N^2) = O(NM)$ (Table 2.11).
In summary, the computational costs of PDM and SPDM on the weakly coupled linear chain model are governed by (a) updating the population $\boldsymbol{n}$ using a sparse stoichiometry representation and (b) never needing to update the partial propensities of unimolecular reactions. Since the linear chain model contains only unimolecular reactions, none of the partial propensities ever need to be updated, leading to an update cost of $\mathcal{C}_{\text{P}} = 0$ (see Table 2.10). While we have implemented SDM according to the original publication [16], we note that if one uses a sparse representation of the stoichiometry matrix also in SDM, point (a) vanishes and $\mathcal{C}_{\mathbf{n}} = 2$ also for SDM. A sparse-stoichiometry SDM would thus have the same scaling of the computational cost on the linear chain model as SPDM, outperforming PDM.

#### 2.4.4.6 Multi-scale biological network: heat-shock response in *Escherichia coli*

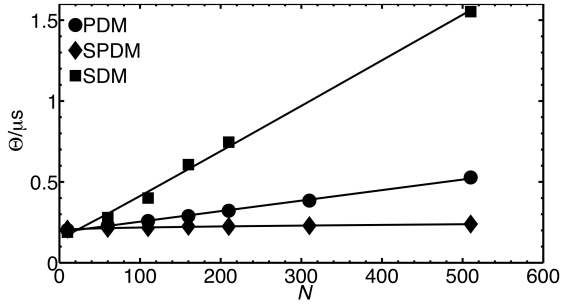
We assess the performance of PDM and SPDM on a small, fixed-size multiscale reaction network. We choose the heat-shock response model since it has also been used to benchmark previous methods, including ODM [15] and SDM [16]. The heat-shock response model [41] was obtained from Dr. Hong Li and Prof. Linda Petzold (UCSB) and is publicly available as part of the StochKit package [42]. The model describes one of the mechanisms used by the bacterium *E. coli* to protect itself against a variety of environmental stresses that are potentially harmful to the structural integrity of its proteins. The heat-shock response (HSR) system reacts to this by rapidly synthesizing heat-shock proteins. The heat-shock sigma factor protein $\sigma^{32}$ activates the HSR by inducing the transcription of heat-shock genes. The heat-shock response model is a small multiscale reaction network (the specific probability rates span 8 orders of magnitude) with $N = 28$ chemical species, $M = 61$ reactions, and a maximum out-degree of the dependency graph of 11 (see Table 2.9). For a detailed description of the model, we refer to Kurata et al. [41]
We simulate the HSR model for $T = 500$ seconds. During this time, approximately 46 million reactions are executed. For a single run, we measure $\Theta^{\text{PDM}} = 0.256\,\mu\text{s}$ and $\Theta^{\text{SDM}} = 0.272\,\mu\text{s}$. This corresponds to a simulated 3.68 million reactions per second of CPU time for SDM and 3.89 million reactions per second for PDM. Hence, PDM is about 6% faster than SDM. This speed-up is mainly due to a smaller $\mathcal{C}_{\text{P}}$ in PDM (see Fig. 2.6(c) for the distribution of updates over all reactions) since the partial propensities of unimolecular reactions never need to be updated. The speed-up, however, is modest because $\mathcal{C}_\mu$ of PDM is $\approx 4.6$ times larger than that of SDM (Table 2.10). This is due to the fact that 95% of all reaction firings are caused by a small subset of only 6 reactions. This multiscale network thus strongly benefits from the dynamic sorting used in SDM. This advantage is recovered in SPDM, where $\mathcal{C}_\mu$ is comparable to that of SDM, and $\Theta^{\text{SPDM}} = 0.245\,\mu\text{s}$ (4.08 million reactions per second). This makes SPDM 11% faster than SDM on this small network.




(a) Colloidal aggregation model

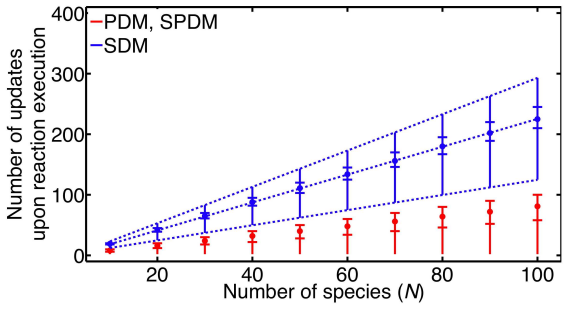



(b) Network of bimolecular reactions

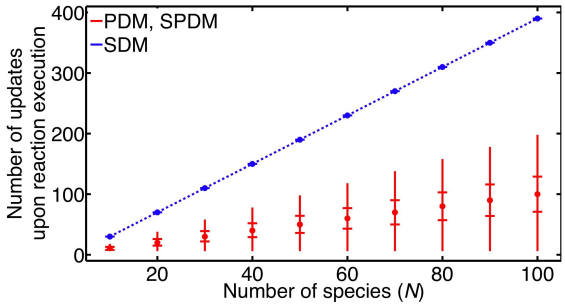



(c) Linear chain model

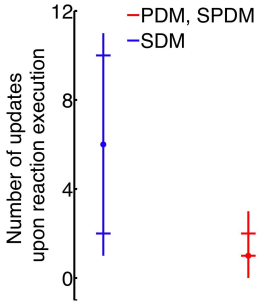
Figure 2.5: Computational costs of PDM (circles), SPDM (diamonds), and SDM (squares). See main text for the simulation parameters and initial conditions used. The average CPU time per reaction (i.e. per time step), $\Theta$, is shown as a function of system size quantified by the number of species $N$. $\Theta$ is defined as the CPU time needed to simulate the system up to final time $T$, divided by the number of reactions executed during this time, and averaged over 100 independent runs (error bars are smaller than symbol size). The solid lines are the corresponding least-squares fits of the scaling $\Theta(N)$ of PDM, SPDM, and SDM with the model $a\mathcal{C}_\mu + b\mathcal{C}_\mathbf{n} + c\mathcal{C}_\mathrm{P}$ on a linear scale (see Table 2.10), where $a$, $b$, and $c$ are the fitted constants. (a) Logarithmic plot of the results for the colloidal aggregation model. The fits are: $\Theta^{\mathrm{PDM}}/\mu\mathrm{s} = 0.0022N + 0.050N^{0.5} + 0.22$, $\Theta^{\mathrm{SPDM}}/\mu\mathrm{s} = 0.0027N + 0.053N^{0.5} + 0.20$, and $\Theta^{\mathrm{SDM}}/\mu\mathrm{s} = 0.00031N^2 + 0.018N + 0.31$. (b) Logarithmic plot of the results for the network of bimolecular reactions. The fits are: $\Theta^{\mathrm{PDM}}/\mu\mathrm{s} = 0.038N$, $\Theta^{\mathrm{SPDM}}/\mu\mathrm{s} = 0.039N$, and $\Theta^{\mathrm{SDM}}/\mu\mathrm{s} = 0.00061N^2 + 0.027N + 0.15$. (c) Linear plot of the results for the linear chain model. The fits are: $\Theta^{\mathrm{PDM}}/\mu\mathrm{s} = 0.00065N + 0.19$, $\Theta^{\mathrm{SPDM}}/\mu\mathrm{s} = 0.0015N^{0.5} + 0.20$, and $\Theta^{\mathrm{SDM}}/\mu\mathrm{s} = 0.0029N - 0.0025N^{0.5} + 0.15$. In all cases, the computational cost $\Theta(N)$ of PDM and SPDM is $O(N)$.




(a) Colloidal aggregation model




(b) Network of bimolecular reactions




(c) Heat shock response model

Figure 2.6: Measured distributions of the number of partial propensities (for PDM and SPDM, red line) or propensities (for SDM, blue line) that need to be updated after firing any reaction of: (a) the colloidal aggregation model, (b) the network of bimolecular reactions, and (c) the heat-shock response model. Dots indicate medians, horizontal bars the upper and lower quartiles, and vertical bars the upper and lower extrema (maximum and minimum). The dotted lines denote the minimum, average and maximum degree of coupling $d_c$ of the reaction networks (see Table 2.9). The number of updates in SDM [16] using a dependency graph is governed by the degree of coupling of the network. In PDM and SPDM, less updates need to be performed since partial propensities depend on the population of at most one species and are constant for unimolecular reactions.

#### 2.4.4.7 Conclusions

When simulating weakly coupled reaction networks, where the maximum number of reactions that are influenced by any reaction is constant with system size, the best computational cost of previous exact SSAs for the sampling step is $O(\log_2 M)$ [1], where $M$ is the total number of reactions, or $O(1)$ under some additional conditions on the propensity distribution [18]. Using dependency graphs, also the update step has been reduced to $O(1)$ for weakly coupled networks [15, 16, 18]. For strongly coupled reaction networks, where the degree of coupling increases with system size and can be as large as the total number of reactions, all previous exact SSA formulations have a computational cost that is $O(M)$.

We have introduced a new quantity called *partial propensity* and have used it to construct two novel formulations of the exact SSA: PDM and its sorting variant SPDM. Both are algebraically equivalent to DM and yield the same population trajectories $\boldsymbol{n}(t)$ as to those produced by DM. In our formulation of partial propensities, we have limited ourselves to elementary chemical reactions. Since their partial propensities depend on the population of at most one species, both new SSA formulations have a computational cost that scales at most linearly with the number of species rather than the number of reactions, independently of the degree of coupling. This is particularly advantageous in strongly coupled reaction networks, where the number of reactions $M$ grows faster than the number of species $N$ with system size. For networks of fixed size, PDM and SPDM are especially efficient when $M \gg N$. PDM's computational cost is $O(N)$, which is made possible by appropriately grouping the partial propensities in the sampling step and formulating a dependency graph over species rather than reactions in the update step. Moreover, the partial propensities of unimolecular reactions and source reactions are constant and never need to be updated. This further reduces the size of the dependency graph and the computational cost of the update step. To our knowledge, PDM is the first SSA formulation that has a computational cost that is $O(N)$, irrespective of the degree of coupling of the reaction network. In the case of multiscale networks, the absolute computational cost of SPDM is smaller than that of PDM.

We have benchmarked PDM and SPDM on four test cases with various degrees of coupling. The first two benchmarks considered strongly coupled networks, where the degree of coupling scales proportionally to the number of species. The third benchmark considered the most weakly coupled network possible, where several other SSA formulations might be more efficient. Finally, the fourth benchmark considered a small biological multiscale network. These benchmarks allowed estimating the scaling of the computational cost with system size and the cost contributions from reaction sampling, population update, and partial-propensity update. The results showed that (a) the overall computational costs of PDM and SPDM are $O(N)$, even for strongly coupled networks, (b) on very weakly coupled networks, SPDM is competitive compared to SDM, (c) on multiscale networks SPDM outperforms PDM, and (d) the memory requirements of PDM and SPDM are $O(M)$ in all cases, and hence not larger than those of any other exact SSA formulation.

PDM and SPDM, however, have a number of limitations. The most important limitation is that the presented formulation of partial propensities is only applicable to elementary chemical reactions. Higher-order chemical reaction can be broken down into elementary reactions at the expense of increasing system size (see Appendix **??**). In applications such as population ecology or social science, the idea of partial propensities can, however, only be used if the (generalized) reactions are at most binary and one species can be factored out, i.e., if the propensity for every reaction between species $\mathrm{S}_i$ and $\mathrm{S}_j$ can be written as $a_\mu = c_\mu n_i \tilde{h}(n_j)$. Besides this structural limitation, the computational performance of the particular algorithms presented here can be limited in several situations. One of them is the simulation of very small networks, where the overhead of the data

structures involved in PDM and SPDM may not be amortized by the gain in efficiency and a simulation using DM may be more efficient. In multiscale networks, where the propensities span several orders of magnitude, PDM is slower than SPDM. In multiscale networks where a small subset ($\ll N$) of all reactions accounts for almost all of the reaction firings, however, the overhead of the data structures involved in SPDM, including their initialization, may not be amortized by the gain in efficiency. Finally, PDM and SPDM were designed to have a computational cost that scales linearly with the number of species rather than the number of reactions. For reaction networks in which the number of reactions grows sub-linearly with the number of species, this becomes a disadvantage. In such cases, SSA formulations that scale with the number of reactions are favorable. Taken together, our results suggest that PDM and SPDM can potentially offer significant performance improvements especially in strongly coupled networks, including the simulation of colloidal aggregation [35, 36, 37, 38, 39], Becker-Döring-like nucleation-and-growth reactions [43], and scale-free biochemical reaction networks, where certain hubs are strongly coupled [44, 45, 46, 40].

### 2.4.5 The partial-propensity SSA with composition-rejection sampling (PSSA-CR)

PDM has a computational cost of $O(N)$ irrespective of the degree of coupling of the reaction network. However, in practice, and especially for networks of fixed size, it is often difficult to determine which coupling class a reaction network belongs to. This is because the coupling class is defined as a function of network size. For fixed-size systems, however, only a single point of that function is known, requiring additional knowledge to determine the coupling class. There is thus a need for an exact SSA that combines the favorable scaling of the computational cost of SSA-CR (Sec. 2.3.7) for weakly coupled networks and of PDM for strongly coupled ones. Here, we use the concept of partial propensities (see Sections 2.4.1 and 2.4.2) to construct a partial-propensity variant of SSA-CR, called the partial-propensity SSA with composition-rejection sampling (PSSA-CR). We show that PSSA-CR has a computational cost of $O(1)$ for weakly coupled networks and $O(N)$ for strongly coupled networks, thus combining the advantages of PDM and SSA-CR.
The partial-propensity SSA with composition-rejection sampling (PSSA-CR) is based on the idea of factorizing the reaction propensities into partial-propensities, grouping and binning them, and using composition-rejection (CR) sampling [30, 18] to determine the index of the next reaction (see Sec. 2.3.7 for the principle behind CR sampling). PSSA-CR reduces the computational cost for weakly coupled reaction networks to $O(1)$ under the assumption that the ratio of maximum to minimum non-zero propensity is bounded by a constant. It achieves this superior scaling for weakly coupled networks while maintaining the computational cost for strongly coupled reaction networks at $O(N)$.

#### 2.4.5.1 Detailed description

PSSA-CR uses a composition-rejection sampling strategy over partial propensities in order to sample the index of the next reaction. Since every reaction in a partial-propensity method is identified by its group index and its element index, we apply two composition-rejection steps: one to sample the group index and one to sample the element index. Table 2.12 gives an overview of PSSA-CR. The individual steps are described in detail below.
The principle data structures in PSSA-CR are the same as in PDM. The partial propensities are stored in a partial-propensity structure $\mathbf{\Pi} = \{\mathbf{\Pi}_i\}_{i=0}^{N}$ as a one-dimensional array of one-dimensional

arrays. The reaction indices $\mu$ corresponding to a certain entry in $\mathbf{\Pi}$ are stored in a look-up table $\mathbf{L} = \{\mathbf{L}_i\}_{i=0}^{N}$, making every reaction $\mu$ identifiable by its group index $I$ and its element index $J$ as $\mu = \mathrm{L}_{I,J}$. The group-sum array $\mathbf{\Lambda}$ stores the sums of the partial propensities in each group $\mathbf{\Pi}_i$, i.e. $\Lambda_i = \sum_j \Pi_{i,j}$. We also store the total propensity of each group in an array $\mathbf{\Sigma}$, computed as $\Sigma_i = n_i \Lambda_i, \quad i = 1, \ldots, N$, and $\Sigma_0 = \Lambda_0$. See Sec. 2.4.3 for more details on the data structures.

In PSSA-CR, the entries of $\mathbf{\Sigma}$ are then sorted into $G_\Sigma = \log_2 \frac{\Sigma_{\max}}{\Sigma_{\min}} + 1$ bins such that bin $b$ contains all $\Sigma_i$'s with $2^{b-1}\Sigma_{\min} \leq \Sigma_i < 2^b \Sigma_{\min}$. $\Sigma_{\min}$ and $\Sigma_{\max}$ are the smallest and largest non-zero values in $\mathbf{\Sigma}$ that can possibly occur during a simulation. They are determined as outlined below. The total propensity of each bin $b$, $\sigma_b^{(\Sigma)}$, is computed by summing up the $\Sigma_i$'s in that bin. Similarly, the entries of each $\mathbf{\Pi}_i$ are sorted into $G_{\Pi_i} = \log_2 \frac{\Pi_{i,\max}}{\Pi_{i,\min}} + 1$ bins with bin $b$ containing all elements in $\mathbf{\Pi}_i$ with $2^{b-1}\Pi_{i,\min} \leq \Pi_{i,j} < 2^b \Pi_{i,\min}$. $\Pi_{i,\min}$ and $\Pi_{i,\max}$ are the smallest and largest non-zero values in $\mathbf{\Pi}_i$ that can possibly occur during a simulation. The total partial propensity of bin $b$ is stored in $\sigma_b^{(\Pi_i)}$. The $\Pi_{i,\min}$'s and $\Sigma_{\min}$ can always be computed *a priori*. $\Pi_{i,\min}$ is the minimum non-zero value in $\mathbf{\Pi}_i$ when all partial propensities are calculated with one molecule of each reactant. $\Sigma_{\min}$ is the minimum among all $n_i \Pi_{i,\min}$'s, where $n_i$ is the population of species $\mathrm{S}_i$ used to calculate $\Pi_{i,\min}$. Estimating the $\Pi_{i,\max}$'s and $\Sigma_{\max}$ *a priori* may be possible using prior knowledge about the chemical reaction network, such as physical constraints. In cases where the $\Pi_{i,\max}$'s and $\Sigma_{\max}$ cannot be estimated *a priori*, PSSA-CR dynamically updates the $\Pi_{i,\max}$'s and $\Sigma_{\max}$ over the course of the simulation. If this increases any $G_{\Pi_i}$ or $G_\Sigma$, the corresponding data structures are dynamically enlarged.

We apply the composition-rejection sampling strategy [30, 18] to obtain the group index $I$ and the element index $J$ of the next reaction $\mu$. The group index $I$ is sampled in two steps: (1) the composition step to find the bin $b_I$ and (2) the rejection step to find $\Sigma_I$ inside that bin. The composition step is done by linear search, thus:

$$b_I = \min \left[ b \ : \ r_1 a < \sum_{i=1}^{b} \sigma_i^{(\Sigma)} \right], \tag{2.42}$$

where $a$ is the total propensity of all reactions in the network and $r_1$ is a uniform random number in $[0, 1)$. The rejection step samples the group-index $I$ from the elements in bin $b_I$. For this step, we generate a uniformly distributed random number $r_2$ in $[0,\ 2^{b_I}\Sigma_{\min})$ and a uniformly distributed random integer $r_3$ between 1 and the number of elements in bin $b_I$. If the $r_3^{\text{th}}$ $\Sigma_i$ in bin $b_I$ is less than $r_2$, the index of that $\Sigma_i$ is chosen as the group-index $I$. If this inequality is not satisfied, the rejection step is repeated. This is illustrated in Fig. 2.7 for an example with 6 partial-propensity groups. Assume that, in this example, the composition step has selected bin $b_I = 2$ as the one containing $\Sigma_I$. The rejection step then samples uniformly random points inside the rectangle defining this bin's value range (bold rectangle). A sample is accepted if it falls inside one of the bars representing the $\Sigma_i$'s. If the first sample (point A in Fig. 2.7 with $r_3 = 2$ and $r_2 > \Sigma_4$) is rejected, sampling is repeated until the point falls inside one of the bars (point B in Fig. 2.7 with $r_3 = 1$ and $r_2 < \Sigma_0$). This determines the group index of the next reaction ($I = 0$ in the example in Fig. 2.7). By binning the $\Sigma_i$'s as described, we ensure that the area covered by the $\Sigma_i$ bars in any bin is at least 50% of the total area of the bin's bounding rectangle. The expected number of iterations of the rejection sampling is hence less than or equal to two.

In order to sample the element index $J$, the same composition-rejection procedure is also applied within the identified group $I$. The composition step again involves a linear search for the bin $b_J$

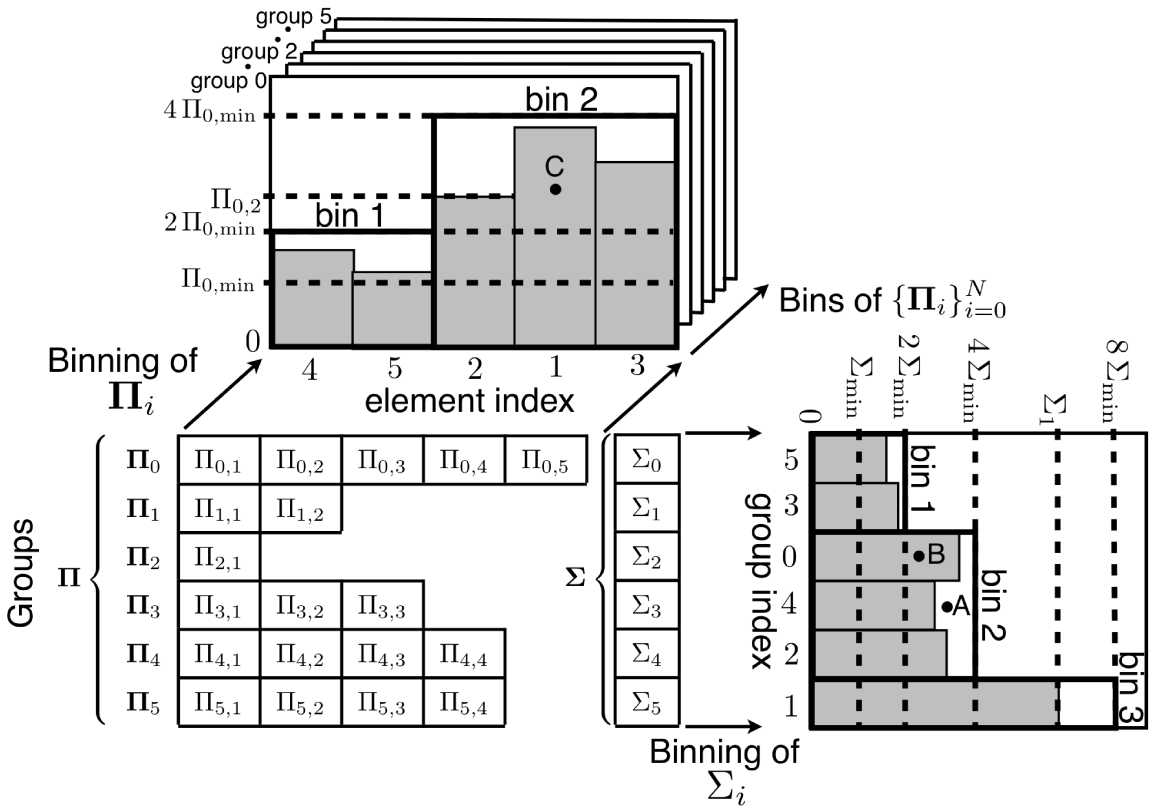



Figure 2.7: Illustration of the two composition-rejection sampling steps used in PSSA-CR. The example shown is for a network with $N = 5$ species and $M = 19$ reactions. The partial propensities are grouped into 6 ($=N+1$) groups in the partial-propensity structure $\{\mathbf{\Pi}\}_{i=0}^{N}$. The sum of propensities in group $\mathbf{\Pi}_i$ is stored in $\Sigma_i$. The elements of $\mathbf{\Sigma}$ and of each $\mathbf{\Pi}_i$ are sorted into dyadic bins. The shaded bars represent the values of the corresponding entries. The extent of each bin is shown by a bold rectangle. Due to the dyadic binning, the bars always cover at least 50% of the area of any bin's rectangle. In order to sample the index of the next reaction, two composition-rejection sampling steps are used: one for the group index $I$ and another one for the element in index $J$ in $\mathbf{\Pi}$. Points A, B, and C refer to the example given in the main text.

containing the partial propensity of the next reaction, as:

$$b_J \;=\; \min\left[b \;:\; r_4\Lambda_I \;<\; \sum_{i=1}^{b} \sigma_i^{(\Pi_I)}\right], \tag{2.43}$$

where $r_4$ is a uniform random number in [0,1). The rejection step as described above is subsequently used to find the element index $J$ from a uniformly distributed random number $r_5$ in $[0,\, 2^{b_J}\Pi_{I,\min})$ and a uniformly distributed random integer $r_6$ between 1 and the number of elements in bin $b_J$. In the example in Fig. 2.7, the group index $I = 0$ has been selected. Assume that the composition step for the element index $J$ has selected bin $b_J = 2$ in the group $\mathbf{\Pi}_0$. Rejection sampling in this bin is then repeated until a point inside any of the bars representing the partial propensities $\Pi_{0,j}$ is selected (point C in Fig. 2.7 with $r_6 = 2$ and $r_5 < \Pi_{0,1}$). This determines the element index of the next reaction ($J = 1$ in the example in Fig. 2.7). The indices $I$ and $J$ of the next reaction are then translated to the reaction index $\mu$ using the look-up table, hence $\mu = \mathrm{L}_{I,J}$.
To execute a sampled reaction, $\boldsymbol{n}$, $\mathbf{\Pi}$, $\mathbf{\Lambda}$, and $\mathbf{\Sigma}$ are updated using the same update algorithm and data structures as in PDM (see Sec. 2.4.3):

$\mathbf{U}^{(1)}$ is a array of $M$ arrays, where the $i^{\text{th}}$ array contains the indices of all species involved in the $i^{\text{th}}$ reaction.

$\mathbf{U}^{(2)}$ is a array of $M$ arrays containing the corresponding stoichiometry (the change in population of each species upon reaction) of the species stored in $\mathbf{U}^{(1)}$.

$\mathbf{U}^{(3)}$ is a array of $N$ arrays, where the $i^{\text{th}}$ array contains the indices of all entries in $\mathbf{\Pi}$ that depend on $n_i$.

After each reaction, we use $\mathbf{U}^{(1)}$ to determine the indices of all species involved in this reaction. The stoichiometry is then looked up in $\mathbf{U}^{(2)}$ and the population $\boldsymbol{n}$ is updated accordingly. Subsequently, $\mathbf{U}^{(3)}$ is used to locate the affected entries in $\mathbf{\Pi}$ and recompute them. Since the partial propensities of unimolecular and source reactions are constant and need never be updated, $\mathbf{U}^{(3)}$ only contains the indices of the partial propensities of bimolecular reactions.
After updating the partial propensities, the bin memberships of all modified $\Pi_{i,j}$'s and $\Sigma_i$'s need to be updated. This requires locating the bin assignment of any $\Pi_{i,j}$ and $\Sigma_i$ in a one-step operation. We implement this by having every $\Pi_{i,j}$ and $\Sigma_i$ store two additional integers: the bin membership and the location inside that bin. Depending on their new value, the changed $\Pi_{i,j}$'s and $\Sigma_i$'s are kept inside the same bin or moved to a different bin. Then, the corresponding bin sums are updated by adding the total change. This can be done in $O(1)$ operations since the ordering of elements in a bin does not matter. Elements that are removed from a bin are simply replaced by the last element in that bin, which is then removed.
The computational cost of PSSA-CR is $O(G_\Sigma + \max\{G_{\Pi_0}, \ldots, G_{\Pi_N}\} + N)$ for strongly coupled reaction networks and $O(G_\Sigma + \max\{G_{\Pi_0}, \ldots, G_{\Pi_N}\})$ for weakly coupled ones (see Sec. 2.4.5.2). If the dynamic range of propensities is bounded over the time of a simulation, the computational cost on weakly coupled networks reduces to $O(1)$ (see Sec 2.4.5.2).
The memory requirement of PSSA-CR is larger than that os PDM. In addition to the data structures required in PDM, PSSA-CR needs an additional $O(N + M)$ memory for the binning of the $\Sigma_i$'s and $\Pi_{i,j}$'s. This renders the memory requirement of PSSA-CR $O(N + M)$, which is equivalent to $O(M)$ since $M$ usually scales faster than $N$ for large reaction networks.

1. Initialize the data structures. Set time $t \leftarrow 0$.
2. While $t < t_{\mathrm{f}}$, where $t_{\mathrm{f}}$ is the final simulation time, repeat:
   2.1. Sample the group index $I$ using composition-rejection sampling.
   2.2. Sample the element index $J$ using composition-rejection sampling.
   2.3. Read the index of the reaction identified by the group index $I$ and the element index $J$ from the look-up table $\mathbf{L}$.
   2.4. Compute the time to the next reaction $\tau \leftarrow a^{-1}\ln(r^{-1})$, where $a$ is the total propensity of all reactions and $r$ a uniformly distributed random number in $[0, 1)$.
   2.5. Update the population of species and the partial propensity structure using the dependency graph over species. Update the bin assignments of changed partial propensities.
   2.6. Increment time: $t \leftarrow t + \tau$.
3. Stop.

Table 2.12: Overview of PSSA-CR.

#### 2.4.5.2 Computational cost

The computational cost of PSSA-CR is determined by the sampling and update steps of the algorithm. Composition-rejection sampling of the group-index $I$ has a cost that is $O(G_\Sigma)$. This is because (a) the composition step involves a linear search over at most $G_\Sigma$ elements and (b) the computational cost of the rejection step is $O(1)$ with increasing network size. The reasoning is as follows: The present binning strategy ensures that at least $1/2$ of the area of each bin is covered by the $\Sigma_i$'s in that bin (see Fig. 2.7). Therefore, the probability of acceptance after $i$ iterations of rejection sampling is $1 - 2^{-i}$. This is independent of network size and hence the rejection step is $O(1)$. The probability of acceptance is at least 0.9999 after 13 iterations of the rejection sampling step*. Likewise, the computational cost of the composition-rejection sampling of the element index $J$ is $O(\max\{G_{\Pi_0}, \ldots, G_{\Pi_N}\})$.
The computational cost of the update step is $O(N)$ like in PDM, albeit with a larger prefactor due to additional overhead associated with reassigning bin memberships. In summary, the total computational cost of PSSA-CR is $O(G_\Sigma + \max\{G_{\Pi_0}, \ldots, G_{\Pi_N}\} + N)$.
For weakly coupled reaction networks, the update step becomes $O(1)$, since the number of entries in $\mathbf{\Pi}$ that need to be updated is independent of system size. This reduces the computational cost of PSSA-CR for weakly coupled networks to $O(G_\Sigma + \max\{G_{\Pi_0}, \ldots, G_{\Pi_N}\})$. In addition, if $\Sigma_{\max}$ and $\Pi_{i,\max}$ are bounded for all $i$, the number of bins $G_\Sigma = \log_2 \frac{\Sigma_{\max}}{\Sigma_{\min}} + 1$ and $G_{\Pi_i} = \log_2 \frac{\Pi_{i,\max}}{\Pi_{i,\min}} + 1$ are also bounded. This renders the computation cost of PSSA-CR $O(1)$ for weakly coupled networks that have a bounded dynamic range of propensities. Even if $G_\Sigma$ and $G_{\Pi_i}$ are not bounded by a constant it is unlikely that they equal to $N$ irrespective of system size. This would require requires that the ratios $\frac{\Sigma_{\max}}{\Sigma_{\min}}$ and $\frac{\Pi_{i,\max}}{\Pi_{i,\min}}$ scale proportionally to $2^N$. In practice we observe that $G_\Sigma$ and

*The number $1 - 2^{-i}$ evaluates to 1 in the standard double-precision representation for $i = 51$.

$G_{\Pi_i}$ scale only weakly with increasing system size $N$. We present this empirical evidence for the weakly coupled cyclic chain model in Sec. 2.4.5.4.

#### 2.4.5.3 Benchmarks

We benchmark the computational performance of PSSA-CR on both a weakly coupled and a strongly coupled reaction network. We choose the cyclic chain model [15, 47] and the colloidal aggregation model [35, 36, 37, 38, 39] as representative networks, respectively. We compare the performance of PSSA-CR with that of SDM, the sorting direct method [16], and SPDM, the analogous sorting variant of PDM (see Sections 2.4.3 and 2.4.4).
All tested SSA formulations are implemented in C++ using the random number generator of the GSL library and compiled using the Intel C++ compiler version 11.1 with the O3 optimization flag. All timings are measured on a Linux 2.6 workstation with a 2.8 GHz quad-core Intel Xeon E5462 processor, 8 GB of memory and 4 MB L2 cache. For all test cases, we simulate the reaction network until $10^7$ reactions have been executed and report the average CPU time $\Theta$ per reaction. All simulations are run without any *a priori* estimate of the $\Pi_{i,\max}$'s and $\Sigma_{\max}$. Instead, the $\Pi_{i,\max}$'s and $\Sigma_{\max}$ are constantly updated over the course of a simulation and the number of bins is dynamically increased when necessary.

#### 2.4.5.4 A weakly coupled reaction network: Cyclic chain model

The cyclic chain model is given by the reaction network

$$\begin{aligned} \mathrm{S}_i &\xrightarrow{c_i} \mathrm{S}_{i+1} \qquad i = 1, \ldots, N-1\,, \\ \mathrm{S}_N &\xrightarrow{c_N} \mathrm{S}_1\,. \end{aligned} \tag{2.44}$$

For $N$ chemical species, this network has $M = N$ reactions. The degree of coupling (maximum out-degree of the dependency graph) of this reaction network is 2, independently of system size.
At time $t = 0$, we set all $n_i = 1$ and all specific probability rates $c_i = 1$. Fig. 2.8A shows $\Theta(N)$ for PSSA-CR, SPDM, and SDM. As expected from the theoretical cost analysis, $\Theta$ is $O(1)$ for PSSA-CR and $O(N)$ for SPDM and SDM. PSSA-CR outperforms SPDM for $N$ above a certain break-even point ($N > 700$ here; Fig. 2.8A) and is faster than SDM for all $N$ tested. Below the break-even point, the overhead of the additional data structures and the binning involved in PSSA-CR is not amortized by the better scaling of the computational cost. The $O(1)$ scaling for PSSA-CR in this case is realized because the reaction network is weakly coupled (degree of coupling is independent of $N$) and all $G_{\Pi_i}$'s and $G_\Sigma$ are constant with system size.
In order to test the efficiency of PSSA-CR for a weakly coupled reaction network with *increasing* number of bins, we simulate this test case with specific probability rates $c_i$ randomly chosen between 1 and $10^6$ from an exponential distribution. All other simulation parameters are unchanged. Fig. 2.8B shows the scaling of $\Theta$ for PSSA-CR, SPDM, and SDM. In this multi-scale case, $G_\Sigma$ increases slowly with system size (by 2% over a 16-fold increase in $N$), leading to a very slow increase in $\Theta$ (proportional to $N^{0.028}$ in this case) of PSSA-CR, as predicted by the theoretical cost analysis. Nevertheless, PSSA-CR is more efficient than SPDM for $N$ above a certain break-even point ($N > 500$ here; Fig. 2.8B) and more efficient than SDM for all $N$ tested.
In summary, the measured computational cost of PSSA-CR is $O(1)$ for the cyclic chain model if the number of bins is bounded. If $G_\Sigma$ or $G_{\Pi_i}$ increase with system size, the computational cost is $O(G_\Sigma + \max\{G_{\Pi_0}, \ldots, G_{\Pi_N}\})$, as shown in Sec. 2.4.5.2.

#### 2.4.5.5 A strongly coupled reaction network: Colloidal aggregation model

The colloidal aggregation model is given in Eq. 2.36. For $N$ chemical species, the number reactions is $M = \left\lfloor \frac{N^2}{2} \right\rfloor$. The degree of coupling of this reaction network is $3N - 7$ and hence scales with system size (Table 2.9).
At time $t = 0$, we set all $n_i = 1$ and all specific probability rates to 1. Fig. 2.8C shows $\Theta(N)$ for PSSA-CR, SPDM, and SDM. $\Theta$ is $O(N)$ for PSSA-CR and SPDM, and it is $O(N^2)$ for SDM. The $\Theta$ of PSSA-CR is always larger than that for SPDM. This constant offset is caused by the additional overhead of binning and bin reassignments in PSSA-CR, which is not necessary in SPDM. The break-even point of PSSA-CR with SDM is around $N > 160$. For systems larger than this, the extra overhead in PSSA-CR is amortized.

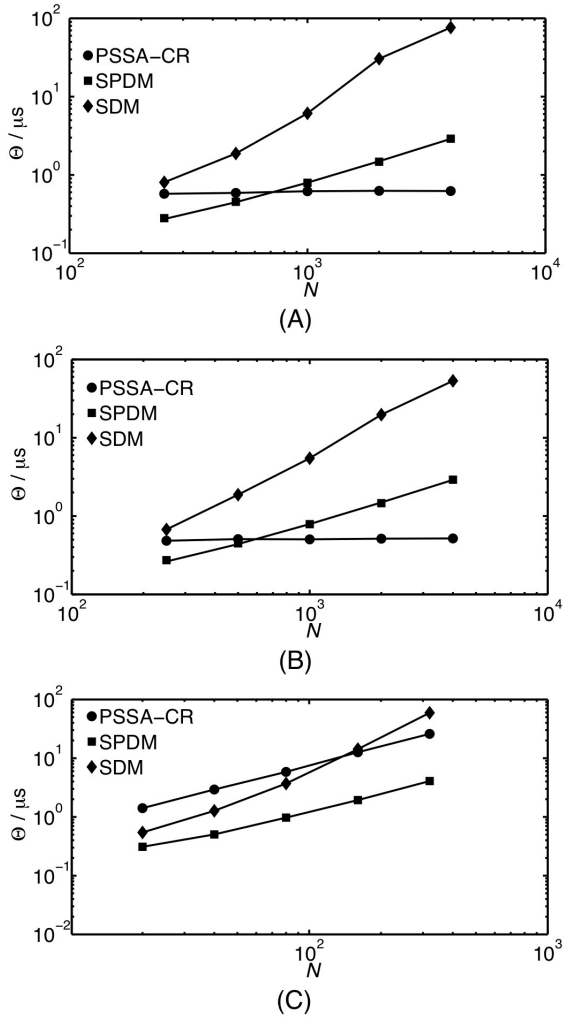



Figure 2.8: Computational cost of PSSA-CR (circles), SPDM (squares), and SDM (diamonds). The average CPU time $\Theta$ per reaction, averaged over 100 independent runs, is shown as a function of the number of species $N$. (A) $\Theta(N)$ for the weakly coupled cyclic chain model with bounded dynamic range of non-zero reaction propensities. $\Theta$ is $O(1)$ for PSSA-CR and $O(N)$ for SPDM and SDM. (B) $\Theta(N)$ for the weakly coupled cyclic chain model with increasing dynamic range of non-zero reaction propensities. $\Theta \propto N^{0.028}$ for PSSA-CR and $\Theta \propto N^1$ for SPDM and SDM. (C) $\Theta(N)$ for the strongly coupled colloidal aggregation model. $\Theta$ is $O(N)$ for both PSSA-CR and SPDM, whereas it is $O(N^2)$ for SDM.

#### 2.4.5.6 Conclusions

We have introduced PSSA-CR, a partial propensity variant of the stochastic simulation algorithm with composition-rejection sampling (SSA-CR) [18]. PSSA-CR uses two composition-rejection sampling steps over partial propensities in order to determine the index of the next reaction. Computational efficiency is achieved by grouping the partial propensities and using dyadic binning in the sampling.
PSSA-CR is an exact SSA formulation whose computational cost is $O(N)$ for strongly coupled reaction networks and $O(1)$ for weakly coupled networks with a bounded range of propensities. We have presented a theoretical cost analysis of PSSA-CR and benchmarked it on three prototypical test cases: (1) a non-stiff weakly coupled reaction network, (2) a multi-scale (stiff) weakly coupled reaction network, and (3) a strongly coupled reaction network. All benchmarks confirmed the theoretically predicted scaling of the computational cost.
PSSA-CR, however, inherits the limitations of PDM (Sec. 2.4.4.7) and of SSA-CR [18]. For small networks, PSSA-CR is outperformed by other methods due to the additional overhead involved in the composition-rejection sampling. SSA formulations such as SDM [16], NRM [1], SSA-CR [18], PDM, or SPDM (see Sections 2.4.3 and 2.4.4) might be more efficient here. In addition, PSSA-CR only achieves the $O(1)$ scaling for weakly coupled networks for which the ratio of maximum to minimum non-zero reaction propensity is bounded by a constant throughout a simulation.
To our knowledge, PSSA-CR has the best scaling of the computational cost on any class of reaction networks. This, however, does not imply that the actual computational cost of PSSA-CR is lowest in all cases, since the pre-factor depends on the data structures involved. If the coupling class of a particular network is not known in practice, however, PSSA-CR seems a reasonable choice for exact stochastic simulations of large reaction networks. Compared to other partial propensity methods, such as SPDM, the better computational scaling of PSSA-CR for weakly coupled networks is paid for by a larger pre-factor in the computational cost for strongly coupled networks.

### 2.4.6 The family of partial-propensity methods

We present the different partial propensity methods like PDM, SPDM, PSSA-CR, dPDM, dSPDM and dPSSA-CR as realizations of a fixed set of modules. We show that by modifying these modules one can flexibly obtain different partial-propensity formulations, each of which being particularly efficient on a certain class of reaction networks. For example, on weakly coupled reaction networks, the partial propensity SSA with composition-rejection sampling (PSSA-CR) has a computational cost of $O(1)$ under the assumption that the ratio of maximum to minimum non-zero propensity is bounded by a constant. On strongly coupled reaction networks, the partial propensity direct method (PDM) is particularly efficient with a computational cost of $O(N)$. On multi-scale strongly coupled networks, the sorting variant of PDM (SPDM) is recommended. For networks with delays, dSPDM is efficient for strongly coupled networks and for weakly coupled networks dPSSA-CR (the delay variant of PSSA-CR) is efficient.

#### 2.4.6.1 Modules of partial-propensity algorithms

The use of partial propensities can be interpreted as follows: Let $\boldsymbol{X}$ be the diagonal matrix of the population vector $\boldsymbol{n}$, such that $\boldsymbol{X} = \mathrm{diag}(\boldsymbol{n})$. Further, let $\boldsymbol{B}$ be the symmetric, positive definite $N \times N$ matrix of specific probability rates of all bimolecular reactions. Element $B_{i,j} = B_{j,i} > 0$ is the specific probability rate $c$ of the reaction of species $i$ with species $j$. Similarly, the specific

probability rates of all unimolecular reactions are collected in the $N \times N$ diagonal matrix $\boldsymbol{U}$. The propensities of bimolecular reactions are then given by the product $\boldsymbol{A}_{\mathrm{B}} = \boldsymbol{X}\boldsymbol{B}\boldsymbol{X}$, those of unimolecular reaction by $\boldsymbol{A}_{\mathrm{U}} = \boldsymbol{X}\boldsymbol{U}$. Traditional SSA formulations amount to first explicitly computing all propensities and then sampling over all the elements in $\boldsymbol{A}_{\mathrm{B,U}} = [\boldsymbol{A}_{\mathrm{B}}, \boldsymbol{A}_{\mathrm{U}}]$. Partial-propensity methods first sample over the vector $\boldsymbol{A}_{\mathrm{B,U}}\,\mathbf{1}$ to obtain the group index $I$, where $\mathbf{1}$ is a vector of 1's. Subsequently sampling the element index $J$ is performed over the elements of the $I^{\mathrm{th}}$ row of the matrix $\boldsymbol{X}^{-1}\boldsymbol{A}_{\mathrm{B,U}} = [\boldsymbol{B}\boldsymbol{X}, \boldsymbol{U}]$. This is implemented using three algorithmic modules: grouping the partial propensities, sampling the next reaction, and updating the values. These modules of partial propensity SSAs are summarized in Fig. 2.9 together with their respective computational costs. Different partial-propensity methods with different computational costs can be constructed by using different algorithms in the sampling module.

1. **Grouping module**: Partial-propensity methods group the partial propensities of all reactions according to the index of the factored-out reactant, i.e., the common reaction partner. Each group thus contains the partial propensities of all reactions having this species as a reactant. The different partial propensities within a group correspond to the various possible reaction partners of the common, factored-out reactant. For any reaction network, there are at most $N+1$ groups (including group 0 for source reactions) and the number of partial propensities in each group is at most $O(N)$. For higher-order reactions (trimolecular and more), multi-dimensional grouping can be used with one dimension per reactant. Again, the total number of groups in each dimension is $O(N)$ and the sampling module can be independently applied in each dimension in order to sample the reaction partners.

2. **Sampling module**: The key building block of partial propensity methods is the algorithm used to sample the time to the next reaction is as in DM and the index of the next reaction. Given the grouping of partial propensities, sampling the index of the next reaction involves sampling the index of the group and then the index of the element within that group. Sampling the index of the group amounts to sampling the first reactant of the next reaction. In order to find out which partner this reactant is going to react with, the partial propensity within the group is sampled. For unimolecular and source reactions, the partial propensities are constants and the second step is obsolete.

   All sampling algorithms used in standard SSAs can also be used in partial-propensity methods. Instead of applying them over reactions, however, they are first applied over partial-propensity groups and then over the elements within the selected group. For example, using linear search (as in Gillespie's direct method [2]) leads to a sampling step that is $O(N)$ on all classes of networks. Replacing linear search by composition-rejection sampling [30] reduces the computational cost of the sampling step to $O(1)$. Other sampling strategies, such as search trees or a first-reaction-method-like sampling over the reaction times can also be used straightforwardly. Depending on the sampling strategy and the associated algorithmic overhead, certain partial-propensity formulations are particularly well suited for certain classes of reaction networks. Also for networks with time delays, different algorithms can be combined for sampling the time to the next reaction and the index of the next reaction. The key difference, however, is that the time to the next reaction and the index of the next reaction are not independent random variables, and hence the time to the next reaction needs to be computed first. See Fig. 2.9 for a summary of different algorithms that can be used. Note that the partial-propensity formulations for networks with delays seamlessly reduce to partial-propensity formulations for network without delays when the delay for each reaction is set to 0.

3. **Update module**: After the selected reaction has fired and the populations of the involved species have been updated, the affected partial propensities are recomputed using a dependency graph over species. Since any partial propensity is a function of the population of at most one species, the number of partial propensities to be updated is at most $O(N)$. In weakly coupled reaction networks, the number of partial propensities to be updated is $O(1)$, since the degree of coupling is bounded by a constant. However, depending on the data structures that are used in the sampling module, the computational cost of the update module varies. Figure 2.9 shows the computational cost of the update step depending on the sampling method used.

Using this modular approach, different algorithms can be combined to construct different partial-propensity formulations (see Fig. 2.9). Certain formulations may be well-suited for reaction networks with certain properties. The classification of reaction networks according to their "difficulty", however, is still largely an open question. Besides system size, degree of coupling, and multiscaling (spectrum of time scales), there might also be other network properties that influence the computational cost of the various SSA formulations. Automatized selection of the most efficient SSA formulation for a given network would require both a systematic classification of networks that goes beyond merely classifying networks as being weakly or strongly coupled. In addition, a prediction of the computational cost of SSA formulations based on network properties would be required. This might involve a more detailed cost analysis of the algorithms and a set of standard benchmark problems that are designed to cover a wide range of performance-relevant parameters.
Implementing the generic modules in C++, we have developed the partial-propensity SSA (pSSA) software package for simulating stochastic chemical kinetics of reaction networks with or without delays. pSSA reads the reaction network in the SBML (Systems Biology Markup Language) [48] input format and is equipped with a user-friendly MATLAB interface. Exact stochastic simulation algorithms supported by the pSSA include DM, PDM, SPDM, PSSA-CR, dDM, dPDM, dSPDM and dPSSA-CR. More details on the pSSA software package can be found in Appendix **??**.

### 2.4.7 Summary

We have introduced partial-propensity formulations of Gillespie's exact stochastic simulation algorithm (SSA). All presented partial-propensity formulations sample trajectories from the exact solution of the chemical master equation (CME). In addition, we also presented a partial-propensity formulation of the delay SSA (dSSA) for chemical reaction networks with delays. We showed that all partial-propensity formulations can be composed from three modules: the grouping module, the sampling module and the update module. Different algorithms and data structures can be used in these modules to obtain partial-propensity formulations. These formulations have varying computational cost depending on the algorithms used and on the coupling class of the simulated reaction network.
Limiting ourselves to networks with elementary reactions, all partial-propensity formulations have a computational cost that scales at most linear with the number of chemical species in the reaction network. Partial-propensity formulations are therefore efficient for reaction networks where the number of chemical species is much smaller than the number of chemical reactions. Due to the overhead of the additional data structures, partial-propensity formulations may not be efficient for small reaction networks, where the additional cost from creating and operating on these data structures may not be amortized.

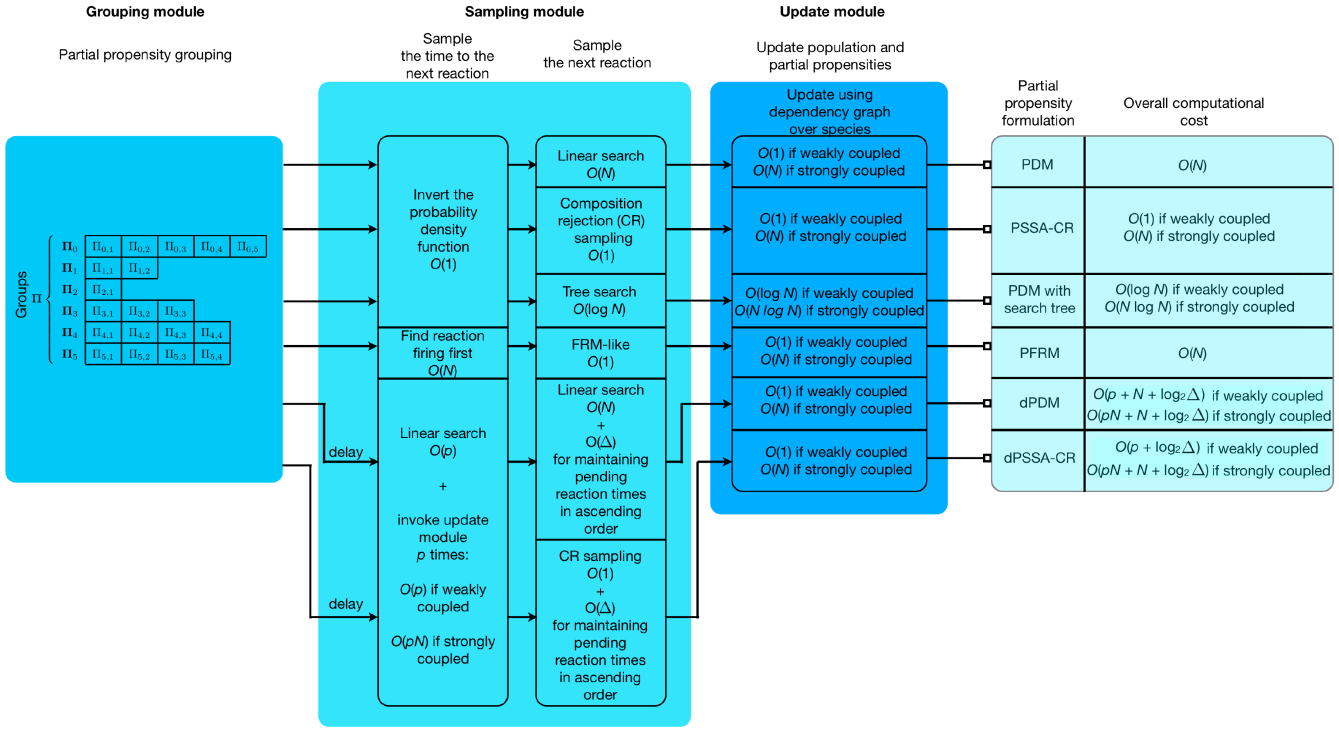



Figure 2.9: Illustration of the three modules of partial propensity SSAs: the grouping module, the sampling module, and the update module. The grouping module groups reactions according to their partial propensities. The sampling module samples the time to the next reaction and the index of the next reaction. The update module updates the population and the affected partial propensities after a reaction has fired. Using different algorithms in the sampling module we obtain different partial propensity formulations, such as PDM (the partial-propensity direct method), PSSA-CR (the partial-propensity SSA with composition-rejection sampling), PDM with search trees, PFRM (the partial-propensity first reaction method), the delay PDM and the delay PSSA-CR. These formulations have varying overall computational costs, depending on the coupling class of the reaction network. $N$ is the number of chemical species in a reaction network, $p$ is the search depth to find the interval for the time to the next reaction is delay SSAs and $\Delta$ is the number of pending reactions in delay SSAs. If the delays of all reactions in a chemical reaction network are zero the computational cost of any delay partial-propensity methods becomes equal to that of the corresponding non-delay formulation. Sorting variants of PDM (SPDM) and of dPDM (dSPDM) is obtained by dynamically bubbling up reactions in the next reaction sampling module.

For strongly and weakly coupled reaction networks without delays, the partial-propensity direct method (PDM) has a computational cost of $O(N)$ where $N$ is the number of chemical species. Due to the dynamic sorting strategy in the sampling module, the sorting variant of PDM (SPDM) is especially efficient for multiscale (stiff) reaction networks, without any significant trade-off in the computational cost for non-multiscale reaction networks. For weakly coupled reaction networks, the computational cost of partial-propensity formulations has been reduced to $O(1)$ using composition-rejection sampling (PSSA-CR). For reaction networks with delays, the delay variants of the these partial-propensity formulations have the same scaling of computational cost with increasing network size.

The favorable scaling of the computational cost of partial-propensity formulations, however, does not necessarily make them the most efficient in terms of absolute runtimes. Even though we have demonstrated that partial-propensity methods can offer significant speed-ups for relatively large reaction networks, this does not imply superior performance on any given, particular network. Based on empirical evidence and on theoretical analysis of the computational costs, however, we recommend SPDM for strongly coupled networks and PSSA-CR for weakly coupled networks. In special cases, SPDM can be worse than PSSA-CR even on strongly coupled reaction networks. If the coupling-class of the reaction network is unknown, we recommend PSSA-CR. For reaction networks with time delay, the corresponding delay variant is recommended.

# 3

# Mesoscopic Modeling and Simulation of Chemical Reaction Networks

*In this chapter:*

- Approximate SSA
- The chemical Kramer-Moyal equation
- The chemical Fokker-Planck equation
- The chemical Langevin equation

*Learning goals:*

- Know how approximate SSA execute more than one reaction at a time
- Know about accuracy and convergence of approximate SSA
- Know and rationalize the assumptions and approximations made when deriving the Fokker-Planck description
- Be able to numerical simulate the Langevin equation

Exact SSA become computationally infeasible when the population of molecules becomes very large, since then $\tau$ goes to zero and time is not advancing. Therefore, coarse-grained descriptions of chemical kinetics are available for the large-population case. There are two ways one can coarse-grain the model: (1) over molecules, (2) over reactions. When coarse-graining over molecules, we do not count every single individual any more in the population, but use a "unit population" to denote several molecules. The population then can also be a non-integer number. In the limit of infinitely large populations, this leads to a continuum of *concentrations*. When coarse-graining over

reactions, one lets go of the requirement to simulate every reaction event, allowing several reactions to fire within any time step. The population, however, remains a vector of integers where every molecule counts. This then leads to *approximate SSAs*.

## 3.1 Approximate Stochastic Simulation Algorithms

In approximate SSA formulations, the population $\boldsymbol{n}(t)$ is described by the equation of motion of a jump Markov process given by

$$\boldsymbol{n}(t+\Delta t) = \boldsymbol{n}(t) + \boldsymbol{\Xi}(\Delta t\,;\,\boldsymbol{n}(t)). \tag{3.1}$$

The random variable $\boldsymbol{\Xi}(\Delta t\,;\,\boldsymbol{n}(t))$, the Markov propagator, can be written as

$$\boldsymbol{\Xi}(\Delta t\,;\,\boldsymbol{n}(t)) = \boldsymbol{\nu}\begin{bmatrix} \psi_1(\Delta t\,;\,\boldsymbol{n}(t)) \\ \vdots \\ \psi_\mu(\Delta t\,;\,\boldsymbol{n}(t)) \\ \vdots \\ \psi_M(\Delta t\,;\,\boldsymbol{n}(t)) \end{bmatrix}, \tag{3.2}$$

where $\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t)) \in \mathbb{Z}_0^+$ is a random variable for the number of times reaction $\mu$ fires in the time interval $[t, t+\Delta t]$. Deriving an exact expression for $\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t))$ is equivalent to solving the CME analytically. Hence, we assume that the propensities $a_\mu$ do not change in $[t, t+\Delta t]$ leading to an approximate solution of the CME. We start by dividing the time interval $[t, t+\Delta t]$ into $k$ equisized subintervals. The probability $P(\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t)) = \lambda)$ that reaction $\mu$ fires in each of $\lambda < k$ subintervals, and does not fire in any of the remaining subintervals, is given by (using Eq. 2.15)

$$\begin{aligned} & P(\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t)) = \lambda) \\ = & \lim_{k\to\infty} \frac{k!}{\lambda!(k-\lambda)!}\left[a_\mu(\boldsymbol{n})\frac{\Delta t}{k} + O\left(\frac{\Delta t^2}{k^2}\right)\right]^\lambda \left[1 - a_\mu(\boldsymbol{n})\frac{\Delta t}{k} + O\left(\frac{\Delta t^2}{k^2}\right)\right]^{(k-\lambda)}. \end{aligned} \tag{3.3}$$

The prefactor is the binomial coefficient of all sets of $\lambda$ subintervals out of $k$. Noting that $\lim_{k\to\infty} O\left(\frac{\Delta t^2}{k^2}\right) = 0$, we get

$$\begin{aligned} P(\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t)) = \lambda) &= \lim_{k\to\infty} \frac{k!}{\lambda!(k-\lambda)!}\left(a_\mu(\boldsymbol{n})\frac{\Delta t}{k}\right)^\lambda \left(1 - a_\mu(\boldsymbol{n})\frac{\Delta t}{k}\right)^{(k-\lambda)} \\ &= \lim_{k\to\infty} \frac{k!}{k^\lambda(k-\lambda)!}\frac{(a_\mu(\boldsymbol{n})\Delta t)^\lambda}{\lambda!}\left(1 - a_\mu(\boldsymbol{n})\frac{\Delta t}{k}\right)^{(k-\lambda)} \\ &= \lim_{k\to\infty} \frac{(k-\lambda+1)\ldots k}{k^\lambda}\frac{(a_\mu(\boldsymbol{n})\Delta t)^\lambda}{\lambda!}\left(1 - a_\mu(\boldsymbol{n})\frac{\Delta t}{k}\right)^{(k-\lambda)} \\ &= \lim_{k\to\infty} \left(1 - \frac{\lambda+1}{k}\right)\ldots\left(1 - \frac{1}{k}\right)\frac{(a_\mu(\boldsymbol{n})\Delta t)^\lambda}{\lambda!}\left(1 - a_\mu(\boldsymbol{n})\frac{\Delta t}{k}\right)^{(k-\lambda)} \\ &= \lim_{k\to\infty} \frac{(a_\mu(\boldsymbol{n})\Delta t)^\lambda}{\lambda!}\left(1 - a_\mu(\boldsymbol{n})\frac{\Delta t}{k}\right)^{k}\left(1 - a_\mu(\boldsymbol{n})\frac{\Delta t}{k}\right)^{-\lambda} \\ &= \frac{(a_\mu(\boldsymbol{n})\Delta t)^\lambda}{\lambda!}\mathrm{e}^{-a_\mu(\boldsymbol{n})\Delta t}. \end{aligned} \tag{3.4}$$

Where in the last step, we have used the definition of the exponential function $\mathrm{e}^x = \lim_{k\to\infty}\left(1+\frac{x}{k}\right)^k$. Therefore, the random variable $\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t))$ is distributed according to the Poisson distribution i.e., $\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t)) \sim \mathcal{P}(a_\mu(\boldsymbol{n})\Delta t)$. In deriving this we assumed that the propensity $a_\mu(\boldsymbol{n})$ does not change during the time interval $[t, t+\Delta t]$. This condition can only be satisfied if exactly one reaction fires per time step just as in exact SSA formulation (see Sec. 2.3). Approximate SSA's impose two conditions [49, 19] :

1. $\Delta t$ must be small enough for the change in the reaction propensities to be small i.e.

$$a_\mu(\boldsymbol{n}(t+\Delta t)) \approx a_\mu(\boldsymbol{n}(t)) \quad \forall\mu. \tag{3.5}$$

2. $\Delta t$ must be large enough for the average number of firings of reaction $\mu$ to be much larger than 1. Otherwise the method offers no improvements of the computational performance over exact SSA formulations.

$$\langle\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t))\rangle = a_\mu(\boldsymbol{n})\Delta t > 1. \tag{3.6}$$

There are various heuristics for choosing a time step $\Delta t$ that satisfies the above two conditions [21, 49, 19]. Note that the second condition is not necessary, but it improves the computational performance of approximate SSA's. In general, since many reactions fire within a time step $\Delta t$ by sampling $\psi_\mu(\Delta t\,;\,\boldsymbol{n}(t))$ from $\mathcal{P}(a_\mu(\boldsymbol{n})\Delta t)$ for $\mu = 1,\ldots,M$, approximate SSA's have a computational cost of $O(M)$ per time step. The computational cost per reaction event is $O(M)$ divided by the number of reactions fired during the time step $\Delta t$. This renders the computational cost of approximate SSAs superior to that of exact SSAs. The scaling of the computational cost of approximate SSAs, however, is the same as that of the exact direct method (DM).

Approximate SSA formulations simulate Eq. 3.1 with different numerical schemes, called $\tau$-leaping, R-leaping, K-leaping, etc. They are computationally efficient when the population of species is not small (more than a few hundreds). The weak order of convergence of these methods has been shown to be at least $\frac{1}{2}$ [50, 51] for some moments of the state-probability function. There are, however, still some disagreements regarding the order of convergence [50, 51]. It is also unclear how quantities that contain information on the path of the stochastic process, such as the time-correlation functions and "higher order quantities", such as switching frequencies in multi-stable systems, converge with decreasing time step $\Delta t$ [52].

## 3.2 Mesoscopic Models of Chemical Kinetics

We define the concentration vector of species in a volume $\Omega$ as

$$\boldsymbol{\phi} = \Omega^{-1}\boldsymbol{n}. \tag{3.7}$$

In terms of the concentration $\boldsymbol{\phi}$, and using the definitions in Eqs. 2.3 and 2.6, the propensity $a_\mu$ can be written as

$$\begin{aligned} a_\mu(\boldsymbol{\phi}) &= a_\mu(\boldsymbol{n}) \\ &= \left(\prod_{i=1}^{N} \frac{(n_i - \nu_{i,\mu}^- + 1)(n_i - \nu_{i,\mu}^- + 2)\dots(n_i - 1)n_i}{\nu_{i,\mu}^-!}\right)\left(\frac{k_\mu \prod_{i=1}^N \nu_{i,\mu}^-!}{\Omega^{(\sum_{i=1}^N \nu_{i,\mu}^-)-1}}\right) \\ &= \left(\prod_{i=1}^{N} \frac{(\phi_i - \frac{\nu_{i,\mu}^- - 1}{\Omega})(\phi_i - \frac{\nu_{i,\mu}^- - 2}{\Omega})\dots(\phi_i - \frac{1}{\Omega})\phi_i}{\nu_{i,\mu}^-!}\right)\left(k_\mu \prod_{i=1}^{N} \nu_{i,\mu}^-!\right)\Omega \\ &= T_\mu(\boldsymbol{\phi})\,\Omega. \end{aligned} \tag{3.8}$$

Rewriting Eq. 2.19 in terms of the concentration $\boldsymbol{\phi}$, we get

$$\frac{\partial P(\boldsymbol{\phi},t)}{\partial t} = \Omega\left[\sum_{\mu=1}^{M} T_\mu(\boldsymbol{\phi} - \Omega^{-1}\boldsymbol{\nu}_\mu)\; P(\boldsymbol{\phi} - \Omega^{-1}\boldsymbol{\nu}_\mu, t) - \; P(\boldsymbol{\phi},t)\; \sum_{\mu=1}^{M} T_\mu(\boldsymbol{\phi})\right]. \tag{3.9}$$

### 3.2.1 The chemical Kramer-Moyal equation

Assuming that the population $\boldsymbol{n}$ increases proportionally with reactor volume $\Omega$, such that the concentration $\boldsymbol{\phi}$ is constant, we can treat $\boldsymbol{\phi}$ as a continuous random variable for a sufficiently large $\Omega$. Performing a Taylor series expansion of the right-hand side of Eq. 3.9, we get

$$\begin{aligned} \frac{\partial P(\boldsymbol{\phi},t)}{\partial t} &= \left\{\sum_{m=1}^{\infty}(-1)^m \Omega^{-(m-1)} \sum_{\substack{i_1,\dots,i_N \\ i_1+\dots+i_N=m}} \frac{1}{i_1!\dots i_N!}\frac{\partial^m}{\partial^{i_1}\phi_1\dots\partial^{i_N}\phi_N} \right. \\ &\qquad \left.\left[\sum_{\mu=1}^{M}\prod_{j=1}^{N}\nu_{j,\mu}^{i_j} T_\mu(\boldsymbol{\phi})\right]\right\} P(\boldsymbol{\phi},t) \end{aligned} \tag{3.10}$$

$$= \left\{\sum_{m=1}^{\infty}\frac{(-1)^m}{\Omega^{(m-1)}} \sum_{\substack{i_1,\dots,i_N \\ i_1+\dots+i_N=m}} \frac{1}{i_1!\dots i_N!}\frac{\partial^m}{\partial^{i_1}\phi_1\dots\partial^{i_N}\phi_N} b_{k\,;\,i_1,\dots,i_N}\right\} P(\boldsymbol{\phi},t), \tag{3.11}$$

where $b_{k\,;\,i_1,\dots,i_N}$ is given by

$$b_{k\,;\,i_1,\dots,i_N} = \left[\sum_{\mu=1}^{M}\prod_{j=1}^{N}\nu_{j,\mu}^{i_j} T_\mu(\boldsymbol{\phi})\right] \qquad i_1+\dots+i_N = k. \tag{3.12}$$

Eq. 3.11 is the *chemical Kramer-Moyal equation* [53, 6, 54, 7, 49] describing the time evolution of the state probability function of a continuous-state jump Markov process (recall that the full CME

described a discrete-state jump Markov process). It's a first-order linear ODE and the solution is given by

$$P(\boldsymbol{\phi},t) = \mathrm{e}^{\left\{ t \left[ \sum_{m=1}^{\infty} \frac{(-1)^m}{\Omega^{(m-1)}} \sum\limits_{\substack{i_1,\ldots,i_N \\ i_1+\ldots+i_N=m}} \frac{1}{i_1!\ldots i_N!} \frac{\partial^m}{\partial^{i_1}\phi_1 \ldots \partial^{i_N}\phi_N} b_{k\,;\,i_1,\ldots,i_N} \right] \right\}} P(\boldsymbol{\phi},0), \tag{3.13}$$

where $P(\boldsymbol{\phi},0)$ is the initial condition. The $b_{k\,;\,i_1,\ldots,i_N}$ are related to the $k^{\text{th}}$ jump moments $B_{k\,;\,i_1,\ldots,i_N}$ of the Markov propagator probability distribution $\Pi(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\,\boldsymbol{n},t)$ as

$$\begin{aligned} B_{k\,;\,i_1,\ldots,i_N} &= \lim_{\mathrm{d}t\to 0} \frac{1}{\mathrm{d}t} \sum_{\mu=1}^{M} \prod_{j=1}^{N} \nu_{j,\mu}^{i_j} \Pi(\boldsymbol{\nu}_\mu \mid \mathrm{d}t\,;\,\boldsymbol{n},t) \qquad (3.14) \\ &= \lim_{\mathrm{d}t\to 0} \frac{1}{\mathrm{d}t} \sum_{\mu=1}^{M} \prod_{j=1}^{N} \nu_{j,\mu}^{i_j} [a_\mu(\boldsymbol{n})\mathrm{d}t + O(\mathrm{d}t^2)] \\ &= \sum_{\mu=1}^{M} \prod_{j=1}^{N} \nu_{j,\mu}^{i_j} a_\mu(\boldsymbol{n}) \\ &= \Omega \sum_{\mu=1}^{M} \prod_{j=1}^{N} \nu_{j,\mu}^{i_j} T_\mu(\boldsymbol{\phi}) \\ &= \Omega\, b_{k\,;\,i_1,\ldots,i_N}. \qquad (3.15) \end{aligned}$$

### 3.2.2 The chemical Fokker-Planck equation and the chemical Langevin equation

Ignoring all terms with $m > 2$ in Eq. 3.11 yields the *nonlinear chemical Fokker-Planck equation*, also known as the *generalized diffusion equation* [55, 56, 57, 58, 59, 49, 60]. This equation appropriately describes stochastic chemical kinetics at large-enough $\Omega$ (i.e., at large population $\boldsymbol{n}$), where the jump moments $B_{k\,;\,i_1,\ldots,i_N} = 0$ for $k > 2$. The Markov propagator probability distribution then becomes Gaussian. This truncation transforms the continuous-state jump Markov process describing chemical kinetics at large $\Omega$ into a continuous Markov process at even larger $\Omega$. The *nonlinear chemical Fokker-Planck equation* is not just an arbitrary truncation of the chemical Kramer-Moyal equation, but is substantiated by theoretical reasoning [61, 49, 19]. From the Taylor expansion of the chemical Kramer-Moyal equation, one can rigorously prove that one has to take either the first two terms, or all terms. Any truncation using more than two, but not all, terms leads to unphysical results where probability densities can become negative. This is the famous Pawula Theorem (1967), singling out the Fokker-Planck equations as the most accurate of all possible truncations of the Kramer-Moyal equation. The nonlinear chemical Fokker-Planck equation is an anisotropic inhomogeneous diffusion equation with drift for the probabilities in concentration space. It can be written as

$$\frac{\partial P(\boldsymbol{\phi},t)}{\partial t} = \boldsymbol{\nabla} \cdot \left[ (2\Omega)^{-1} \boldsymbol{D}(\boldsymbol{\phi}) \boldsymbol{\nabla} P(\boldsymbol{\phi},t) - \boldsymbol{F}(\boldsymbol{\phi}) P(\boldsymbol{\phi},t) \right] \tag{3.16}$$

where $\boldsymbol{\nabla} = \left[ \frac{\partial}{\partial \phi_1}, \ldots, \frac{\partial}{\partial \phi_N} \right]^{\mathrm{T}}$. The drift $\boldsymbol{F}$ is the vector of first moments $b_{1\,;\,i_1,\ldots,i_N}$

$$\boldsymbol{F}(\boldsymbol{\phi}) = \boldsymbol{\nu} \boldsymbol{T}(\boldsymbol{\phi}) \tag{3.17}$$

where $\boldsymbol{T}(\boldsymbol{\phi}) = [T_1(\boldsymbol{\phi}), \ldots, T_M(\boldsymbol{\phi})]^{\mathrm{T}}$. The diffusion tensor $\boldsymbol{D}$ is the matrix of second moments $b_{2\,;\,i_1,\ldots,i_N}$ (see Eqs. 3.15 and 3.14) [**?**]

$$\boldsymbol{D}(\boldsymbol{\phi}) = \boldsymbol{\nu}\ \mathrm{diag}(\boldsymbol{T}(\boldsymbol{\phi}))\ \boldsymbol{\nu}^{\mathrm{T}}, \tag{3.18}$$

where

$$\mathrm{diag}(\boldsymbol{T}(\boldsymbol{\phi})) = \begin{bmatrix} T_1(\boldsymbol{\phi}) & 0 & \ldots & 0 \\ 0 & T_2(\boldsymbol{\phi}) & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & T_M(\boldsymbol{\phi}) \end{bmatrix}. \tag{3.19}$$

Under this approximation, the equation of motion of the Markov process can be written as

$$\boldsymbol{\phi}(t + \mathrm{d}t) = \boldsymbol{\phi}(t) + \boldsymbol{\Xi}(\mathrm{d}t\,;\,\boldsymbol{\phi}, t), \tag{3.20}$$

where $\boldsymbol{\Xi}$ is a random variable distributed as $\boldsymbol{\Xi}(\mathrm{d}t\,;\,\boldsymbol{\phi}, t) \sim \Pi(\Delta\boldsymbol{\phi} \mid \mathrm{d}t\,;\,\boldsymbol{\phi}, t)$ with $\Pi(\Delta\boldsymbol{\phi} \mid \mathrm{d}t\,;\,\boldsymbol{\phi}, t)$ the Gaussian distribution $\mathcal{N}(\boldsymbol{F}(\boldsymbol{\phi})\mathrm{d}t, \Omega^{-1}\boldsymbol{D}(\boldsymbol{\phi})\mathrm{d}t)$ [58, 59]. Here $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ denotes a multivariate Gaussian distribution with mean vector $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma}$. Eq. 3.20 is called the *chemical Langevin equation* (CLE) [49]. According to the CLE, $\boldsymbol{\phi}$ can be considered the coordinate of an overdamped particle moving in the $N$-dimensional concentration space under the influence of a force $\boldsymbol{F}(\boldsymbol{\phi})$ (drift) and an anisotropic zero-mean Gaussian perturbation with covariance $\Omega^{-1}\boldsymbol{D}(\boldsymbol{\phi})$ (diffusion) [62, 63].

### 3.2.3 Simulating the chemical Langevin equation

# 4

# Macroscopic Modeling and Simulation of Chemical Reaction Networks

*In this chapter:*

- The reaction rate equation
- Steady-state flux balance analysis
- Exploiting symmetry in interactions

*Learning goals:*

- Know the assumptions and approximations made when deriving the reaction rate equations
- Be able to use flux balance analysis to study the steady state of a reaction network
- Be able to numerically solve reaction rate equations
- Know about convergence and stability of the numerical solution

## 4.1 The Reaction Rate Equations

The function $T_\mu(\boldsymbol{\phi})$ (Eq. 3.8) explicitly depends on $\boldsymbol{\phi}$ and $\Omega$. These dependences can be separated by expanding $T_\mu(\boldsymbol{\phi})$

$$T_\mu(\boldsymbol{\phi}) = \sum_{i=0}^{\infty} \Omega^{-i} T_{\mu,i}(\boldsymbol{\phi}), \tag{4.1}$$

where the $T_{\mu,i}$'s only depend on $\boldsymbol{\phi}$ and not explicitly on $\Omega$. The moments $b_{k\,;\,i_1,\ldots,i_N}$ (Eq. 3.15), the drift vector $\boldsymbol{F}(\boldsymbol{\phi})$ and the diffusion $\boldsymbol{D}(\boldsymbol{\phi})$ in the nonlinear Fokker-Planck equation (Eqs. 3.16, 3.17 and 3.18) can therefore be written as:

$$b_{k\,;\,i_1,\ldots,i_N} = \sum_{i=0}^{\infty} b^{(i)}_{k\,;\,i_1,\ldots,i_N} \Omega^{-i}, \tag{4.2}$$

$$\boldsymbol{F}(\boldsymbol{\phi}) = \sum_{i=0}^{\infty} \boldsymbol{F}^{(i)}(\boldsymbol{\phi}) \Omega^{-i} \tag{4.3}$$

and

$$\boldsymbol{D}(\boldsymbol{\phi}) = \sum_{i=0}^{\infty} \boldsymbol{D}^{(i)}(\boldsymbol{\phi}) \Omega^{-i}. \tag{4.4}$$

The first few $T_{\mu,i}$'s are:

$$T_{\mu,0}(\boldsymbol{\phi}) = k_\mu \prod_{i=1}^{N} \phi_i^{\nu^-_{i,\mu}}, \tag{4.5}$$

$$T_{\mu,1}(\boldsymbol{\phi}) = -k_\mu \left( \prod_{i=1}^{N} \phi_i^{\nu^-_{i,\mu}} \right) \left[ \sum_{j=1}^{N} \frac{\nu^-_{j,\mu}(\nu^-_{j,\mu}-1)}{2\phi_j} \right] = -\sum_{i=1}^{N} \frac{\phi_i}{2} \frac{\partial^2 T_{\mu,0}(\boldsymbol{\phi})}{\partial \phi_i^2} \tag{4.6}$$

and

$$\begin{aligned} T_{\mu,2}(\boldsymbol{\phi}) &= k_\mu \left( \prod_{i=1}^{N} \phi_i^{\nu^-_{i,\mu}} \right) \left[ \sum_{p=1}^{N-1} \sum_{q=p+1}^{N} \frac{\nu^-_{p,\mu}\nu^-_{q,\mu}(\nu^-_{p,\mu}-1)(\nu^-_{q,\mu}-1)}{2\phi_p\phi_q} \right] \\ &+ k_\mu \left( \prod_{i=1}^{N} \phi_i^{\nu^-_{i,\mu}} \right) \left[ \sum_{j=1}^{N} \frac{\nu^-_{j,\mu}(\nu^-_{j,\mu}-1)(\nu^-_{j,\mu}-2)(3\nu^-_{j,\mu}-1)}{12\phi_j^2} \right]. \end{aligned} \tag{4.7}$$

Substituting Eq. 4.1 in Eq. 3.10 and grouping the terms according to powers of $\Omega$, we get

$$\begin{aligned} \frac{\partial P(\boldsymbol{\phi},t)}{\partial t} &= \Omega^0 \left[ -(\boldsymbol{\nu}_\mu \cdot \boldsymbol{\nabla}) T_{\mu,0}(\boldsymbol{\phi}) \right] \\ &+ \frac{\Omega^{-1}}{2} \left[ (\boldsymbol{\nu}_\mu \cdot \boldsymbol{\nabla})^2 T_{\mu,0}(\boldsymbol{\phi}) - 2(\boldsymbol{\nu}_\mu \cdot \boldsymbol{\nabla}) T_{\mu,1}(\boldsymbol{\phi}) \right] P(\boldsymbol{\phi},t) \\ &+ O(\Omega^{-2}) \end{aligned} \tag{4.8}$$

Taking the limit $\lim_{\Omega\to\infty}$ we get

$$\frac{\partial P(\boldsymbol{\phi},t)}{\partial t} = -\left[ (\boldsymbol{\nu}_\mu \cdot \boldsymbol{\nabla}) T_{\mu,0}(\boldsymbol{\phi}) \right]. \tag{4.9}$$

This equation further reduces the continuous-state Markov process described by the chemical non-linear Fokker-Planck equation to a *Liouville* process. Its solution for $P(\boldsymbol{\phi}', t)$ is the Kronecker delta function $\delta(\boldsymbol{\phi}' - \boldsymbol{\phi})$ where the concentration $\boldsymbol{\phi}$ with unit probability is given by the solution of the ordinary differential equation

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \boldsymbol{\nu}\boldsymbol{T}_0(\boldsymbol{\phi}). \tag{4.10}$$

Here $\boldsymbol{T}_0(\boldsymbol{\phi}) = [T_{1,0}(\boldsymbol{\phi}), \ldots, T_{M,0}(\boldsymbol{\phi})]^{\mathrm{T}}$. Eq. 4.10 is the classical *reaction rate equation* (RRE), which can be independently derived from statistical mechanics. The initial condition for the RRE is given by

$$\boldsymbol{\phi}(t = 0) = \boldsymbol{\phi}_0 = \Omega^{-1}\boldsymbol{n}_0, \tag{4.11}$$

where $\boldsymbol{n}_0$ is the initial population (see Eq. 2.22). In some parts of the literature $T_{\mu,0}$ is referred to as the rate function or the flux of reaction $\mu$ since it describes the flux of converting the reactants of reaction $\mu$ into the corresponding products.
The conceptual difference between the CME and its limit case RRE is illustrated in Fig. 4.1.

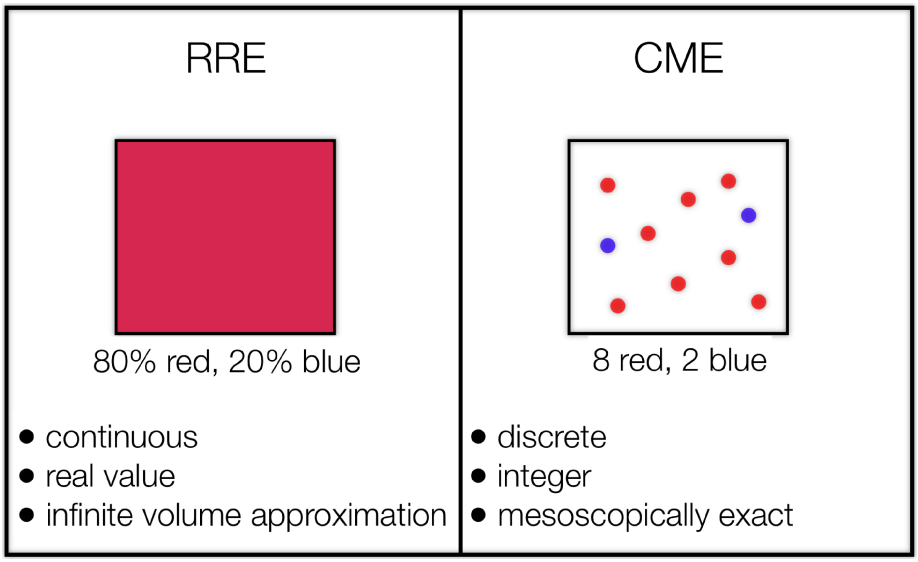



Figure 4.1: Illustration showing the conceptual difference between the reaction rate equation (RRE) and the chemical master equation (CME).

## 4.2 Flux Balance Analysis

Chemical kinetics in the deterministic approximation (see Eq. 4.10) is described by the reaction rate equation (RRE) given by

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \boldsymbol{\nu}\boldsymbol{T}_0(\boldsymbol{\phi}), \tag{4.12}$$

where $\boldsymbol{T}_0(\boldsymbol{\phi})$ is the *flux vector* of all the $M$ reactions and $\boldsymbol{\phi}$ the concentration vector.
The flux balance of the chemical reaction network can be defined by the null space of $\boldsymbol{\nu}$: **The null space of $\boldsymbol{\nu}$** is defined as the space of all flux vectors $\boldsymbol{f}$ for which

$$\boldsymbol{\nu}\boldsymbol{f} = \mathbf{0}. \tag{4.13}$$

This space defines the relationships between the steady-state fluxes of the individual reactions in the network, since Eq. 4.13 implies $\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \mathbf{0}$ (see Eq. 4.12).

### 4.2.1 Examples

#### 4.2.1.1 Cyclic chain model

We again consider the cyclic chain reaction network, which we have already used as an example in Section 1.3:

$$\begin{array}{rcll} \mathrm{S}_i & \rightarrow & \mathrm{S}_{i+1}, & i = 1, \ldots, N-1 \\ \mathrm{S}_N & \rightarrow & \mathrm{S}_1. & \end{array} \tag{4.14}$$

For $N = 3$ the reaction network is

$$\begin{array}{rrcl} \text{Reaction 1}: & \mathrm{S}_1 & \rightarrow & \mathrm{S}_2 \\ \text{Reaction 2}: & \mathrm{S}_2 & \rightarrow & \mathrm{S}_3 \\ \text{Reaction 3}: & \mathrm{S}_3 & \rightarrow & \mathrm{S}_1. \end{array} \tag{4.15}$$

As a reminder, the stoichiometry matrix of this reaction network is:

$$\boldsymbol{\nu} = \boldsymbol{\nu}^+ - \boldsymbol{\nu}^- = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}. \tag{4.16}$$

The null space $\boldsymbol{f}$ of $\boldsymbol{\nu}$ is

$$\boldsymbol{f} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \alpha_1, \tag{4.17}$$

where $\alpha_1$ is any constant. Therefore, the steady-state flux vector can be written as

$$\boldsymbol{T}_0(\phi_{\mathrm{ss}}) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \alpha_1, \tag{4.18}$$

i.e., the steady-state fluxes of all reactions are equal multiples of the constant $\alpha_1$. According to the definition in Eq. 4.5, the fluxes are always non-negative. Therefore, $\alpha_1$ can only be any non-negative constant.

#### 4.2.1.2 Colloidal aggregation model

As a second example, we also consider again the colloidal aggregation model from Section 1.3:

$$\mathrm{S}_i + \mathrm{S}_j \quad \rightarrow \quad \mathrm{S}_{i+j} \qquad i + j = N. \tag{4.19}$$

Species $\mathrm{S}_i$ can be considered a multimer consisting of $i$ monomers.
For $N = 4$ the reaction network is

$$\begin{array}{lrcl} \text{Reaction 1}: & \mathrm{S}_1 + \mathrm{S}_1 & \rightarrow & \mathrm{S}_2 \\ \text{Reaction 2}: & \mathrm{S}_1 + \mathrm{S}_2 & \rightarrow & \mathrm{S}_3 \\ \text{Reaction 3}: & \mathrm{S}_1 + \mathrm{S}_3 & \rightarrow & \mathrm{S}_4 \\ \text{Reaction 4}: & \mathrm{S}_2 + \mathrm{S}_2 & \rightarrow & \mathrm{S}_4. \end{array} \tag{4.20}$$

The stoichiometry matrix of this reaction network is:

$$\boldsymbol{\nu} = \boldsymbol{\nu}^+ - \boldsymbol{\nu}^- = \begin{bmatrix} -2 & -1 & -1 & 0 \\ 1 & -1 & 0 & -2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{4.21}$$

The null space $\boldsymbol{f}$ of $\boldsymbol{\nu}$ is

$$\boldsymbol{f} = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \alpha_1. \tag{4.22}$$

Therefore, the steady-state flux vector can be written as

$$\boldsymbol{T}_0(\phi_{\mathrm{ss}}) = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 0 \end{bmatrix} \alpha_1. \tag{4.23}$$

Again, all steady-state fluxes are multiples of a non-negative constant $\alpha_1$. Since $\alpha_1$ is non-negative, the above relationship indicates that the steady-state fluxes of reactions 2 and 3 in Eq. 4.20 are negative. Since fluxes are always non-negative, this imposes that $\alpha_1 = 0$. Consequently, the above flux relation imposes that all steady-state fluxes in the reaction network of Eq. 4.20 are zero. This can be verified by simulating the RRE with any initial condition and rates. The reasoning can also be as follows: Consider that the initial state of the system is given by 10 molecules of species $\mathrm{S}_1$ and 0 molecules of all other species. Depending on the reaction rates (assuming that they are all non-zero), the steady-state population would be one of two possibilities: 1 molecule of $\mathrm{S}_2$ and 2 molecules of $\mathrm{S}_4$, or 2 molecules of $\mathrm{S}_3$ and 1 molecule of $\mathrm{S}_4$. Both of these population vectors would render all fluxes zero as there would not be enough reactant molecules for any reaction to have a non-zero flux (see Eq. 4.5 in Sec. 4.1 for the definition of reaction fluxes $\boldsymbol{T}_0(\boldsymbol{\phi})$).

## 4.3 Numerical simulation of reaction rate equations

The reaction rate equations are ordinary differential equations describing a smooth continuous-time process $\boldsymbol{\phi}(t)$. In order to simulate this process in the computer, the continuous time derivative $\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}$

needs to be discretized, since digital computers cannot handle continuous mathematical objects. Discretization refers to the process of converting these continuous time derivatives into discrete counterparts by restricting the solution to a finite set of representative time points. This enables evaluation of time derivatives at discrete times $t_n = n\delta t$ where $n = 0, \ldots, N$, $t_n$ is the time stamp at the $n$-th time step and $\delta t$ is referred to as the finite time-step size or the time interval between two consecutive time steps.
Time discretization schemes are referred to as *explicit* if the values of dynamical quantities of interest (here $\boldsymbol{\phi}$) at time $t_n$ can be computed using the values of these quantities at times $t_k$ such that $k < n$. In other words, time discretization scheme are referred to as explicit if the next state of the system can be computed using the current state of the system and the past states of the system. Since this iteratively evolves the system forward in time, the resulting algorithms are also often referred to as *time-stepping* methods. While this is not the only way the reaction rate equations can be discretized (other ways include implicit solvers and spectral Fourier-space methods), we here restrict our discussion to explicit schemes, since they are intuitive and simple to implement.

#### 4.3.0.1 Explicit Euler scheme

Explicit Euler time discretization scheme (also referred to as forward Euler scheme) is the one of the simplest way to discretize time derivatives. In this scheme, time derivative of $\boldsymbol{\phi}$ at time $t_n$ is approximated by the backward finite difference:

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}(t_n) \approx \frac{\boldsymbol{\phi}(t_{n+1}) - \boldsymbol{\phi}(t_n)}{t_{n+1} - t_n} = \frac{\boldsymbol{\phi}(t_{n+1}) - \boldsymbol{\phi}(t_n)}{\delta t}. \tag{4.24}$$

Therefore, the reaction-rate equation $\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}(t) = \boldsymbol{\nu}\boldsymbol{T}_0(\boldsymbol{\phi}) = \boldsymbol{f}(\boldsymbol{\phi}(t))$ is discretized as

$$\begin{aligned} \frac{\boldsymbol{\phi}(t_{n+1}) - \boldsymbol{\phi}(t_n)}{\delta t} &\approx \boldsymbol{f}(\boldsymbol{\phi}(t_n)) \\ i.e., \qquad \boldsymbol{\phi}(t_{n+1}) &\approx \boldsymbol{\phi}(t_n) + \delta t\, \boldsymbol{f}(\boldsymbol{\phi}(t_n)), \qquad n = 0, 1, \ldots. \end{aligned}$$

This scheme is explicit since $\boldsymbol{\phi}(t_{n+1})$ can be computed with the knowledge of $\boldsymbol{\phi}(t_n)$.

**Discretization error.** We analyze the error and the approximation order of discretizing the continuous time derivative using the explicit Euler scheme.
Expanding $\boldsymbol{\phi}(t_n + \delta t)$ around $\delta t$ using a Taylor series expansion assuming $\delta t$ is small and that $\boldsymbol{\phi}(t)$ is an infinitely smooth function in $t$, we observe that

$$\begin{aligned} \boldsymbol{\phi}(t_n + \delta t) &= \boldsymbol{\phi}(t_n) + \frac{\delta t}{1!}\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}(t_n) + \frac{\delta t^2}{2!}\frac{\mathrm{d}^2\boldsymbol{\phi}}{\mathrm{d}t^2}(t_n) + \frac{\delta t^3}{3!}\frac{\mathrm{d}^3\boldsymbol{\phi}}{\mathrm{d}t^3}(t_n) + \ldots \\ &= \boldsymbol{\phi}(t_n) + \frac{\delta t}{1!}\boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)) + \frac{\delta t^2}{2!}\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}t}(t_n, \boldsymbol{\phi}(t_n)) \\ &\quad + \frac{\delta t^3}{3!}\frac{\mathrm{d}^2\boldsymbol{f}}{\mathrm{d}t^2}(t_n, \boldsymbol{\phi}(t_n)) + \ldots. \end{aligned}$$

As $\delta t$ is continuously decreased, the term $\frac{1}{2!}\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}t}(t_n, \boldsymbol{\phi}(t_n))\, O(\delta t^2)$ dominates the error as $\delta t$ approaches 0. Therefore,

$$\boldsymbol{\phi}(t_n + \delta t) = \boldsymbol{\phi}(t_n) + \delta t\, \boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)) + \frac{1}{2!}\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}t}(t_n, \boldsymbol{\phi}(t_n))\, O(\delta t^2). \tag{4.25}$$

This means that the discretization error per time-step using explicit Euler scheme is $O(\delta t^2)$. Therefore, the local order of approximation in an explicit Euler scheme is 2 (the exponent of $\delta t$ in the leading or dominant error term). To compute the the state of the system at time $t = T$, we need to perform $N_T = \frac{T}{\delta t}$ time-steps. The overall discretization error is therefore $N_T O(\delta t^2) = \frac{T}{\delta t} O(\delta t^2) = TO(\delta t) = O(\delta t)$. The global order of approximation of an explicit Euler scheme is 1. The explicit Euler method is therefore said to be first-order accurate. As a consequence, as we half the time-step $\delta t$, the error decreases by a factor of 2.

#### 4.3.0.2 Leapfrog scheme

In a leapfrog scheme, time derivative of $\boldsymbol{\phi}$ at time $t_n$ is approximated by the central finite difference:

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}(t_n) \approx \frac{\boldsymbol{\phi}(t_{n+1}) - \boldsymbol{\phi}(t_{n-1})}{t_{n+1} - t_{n-1}} = \frac{\boldsymbol{\phi}(t_{n+1}) - \boldsymbol{\phi}(t_{n-1})}{2\,\delta t}.$$

The reaction-rate equation $\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}(t) = \boldsymbol{f}(\boldsymbol{\phi}(t))$ is approximated as

$$\begin{aligned} \frac{\boldsymbol{\phi}(t_{n+1}) - \boldsymbol{\phi}(t_{n-1})}{2\delta t} &\approx \boldsymbol{f}(\boldsymbol{\phi}(t_n)) \\ \textit{i.e.,} \qquad \boldsymbol{\phi}(t_{n+1}) &\approx \boldsymbol{\phi}(t_{n-1}) + 2\delta t\, \boldsymbol{f}(\boldsymbol{\phi}(t_n)), \qquad n = 0, 1, \ldots. \end{aligned} \tag{4.26}$$

The leapfrog scheme is therefore explicit but apart from the knowledge of $\boldsymbol{\phi}$ at $t_n$, it also requires the knowledge of $\boldsymbol{\phi}$ at $t_{n-1}$.

**Discretization error.** The leap-frog scheme is given by

$$\boldsymbol{\phi}(t_n + \delta t) \approx \boldsymbol{\phi}(t_n - \delta t) + 2\delta t\, \boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)). \tag{4.27}$$

Expanding $\boldsymbol{\phi}(t_n + \delta t)$ around $\delta t$:

$$\begin{aligned} \boldsymbol{\phi}(t_n + \delta t) &= \boldsymbol{\phi}(t_n) + \frac{\delta t}{1!}\boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)) + \frac{\delta t^2}{2!}\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}t}(t_n, \boldsymbol{\phi}(t_n)) \\ &\quad + \frac{\delta t^3}{3!}\frac{\mathrm{d}^2\boldsymbol{f}}{\mathrm{d}t^2}(t_n, \boldsymbol{\phi}(t_n)) + \frac{\delta t^4}{4!}\frac{\mathrm{d}^4\boldsymbol{f}}{\mathrm{d}t^4}(t_n, \boldsymbol{\phi}(t_n)) + \ldots. \end{aligned}$$

Similarly, expanding $\boldsymbol{\phi}(t_n - \delta t)$ around $\delta t$:

$$\begin{aligned} \boldsymbol{\phi}(t_n - \delta t) &= \boldsymbol{\phi}(t_n) - \frac{\delta t}{1!}\boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)) + \frac{\delta t^2}{2!}\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}t}(t_n, \boldsymbol{\phi}(t_n)) \\ &\quad - \frac{\delta t^3}{3!}\frac{\mathrm{d}^2\boldsymbol{f}}{\mathrm{d}t^2}(t_n, \boldsymbol{\phi}(t_n)) + \frac{\delta t^4}{4!}\frac{\mathrm{d}^4\boldsymbol{f}}{\mathrm{d}t^4}(t_n, \boldsymbol{\phi}(t_n)) + \ldots. \end{aligned}$$

Therefore,

$$\begin{aligned} \boldsymbol{\phi}(t_n + \delta t) - \boldsymbol{\phi}(t_n - \delta t) &= 2\frac{\delta t}{1!}\boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)) + 2\frac{\delta t^3}{3!}\frac{\mathrm{d}^2\boldsymbol{f}}{\mathrm{d}t^2}(t_n, \boldsymbol{\phi}(t_n)) + \ldots \\ &= 2\delta t\, \boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)) + O(\delta t^3). \end{aligned}$$

Equivalently,

$$\boldsymbol{\phi}(t_n + \delta t) \quad = \quad \boldsymbol{\phi}(t_n - \delta t) + 2\delta t\, \boldsymbol{f}(t_n, \boldsymbol{\phi}(t_n)) + O(\delta t^3). \tag{4.28}$$

Comparing Eqs. 4.27 and 4.28, we see that the local discretization error in a leap-frog time-stepping scheme is $O(\delta t^3)$ and therefore the local order of approximation is 3. The global discretization error is $O(\delta t^2)$ and therefore the global order of approximation is 2. Leap-frog time-stepping scheme is therefore a second-order accurate time discretization scheme. As a consequence, as we half the time-step $\delta t$, the error decreases by a factor of 4. In the next section, we present a scheme that is fourth-order accurate in which halving $\delta t$ results in the error decreasing by a factor of 16.

### 4.3.1 Numerical stability

In the previous section, we learned about different time-stepping schemes and the discretization error of these schemes. Here, we analyze the numerical stability of these schemes. Numerical stability is a property desired by all time-stepping schemes. This property requires that when an underlying mathematical equation being discretized is bounded for all times, then the corresponding discretization scheme of the underlying equations also results in bounded values for the dynamical variables at all times. We will analyze the numerical stability for a linear right-hand side:

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = \lambda\, x(t), \tag{4.29}$$

where $\lambda = \lambda_{\mathrm{R}} + i\,\lambda_{\mathrm{I}}$ is in general complex with $\lambda_{\mathrm{R}}$ being the real part and $\lambda_{\mathrm{I}}$ being the imaginary part. Closed-form stability analysis is only possible for linear right-hand sides, i.e., linear flux vectors. The famous Strang Theorem, however, guarantees that if the time-stepping scheme is stable for a linear right-hand side, it is also stable for nonlinear right-hand sides. The reverse is not true (i.e., instability in the linear case does not imply instability in the nonlinear case). The solution of this equation is

$$\begin{aligned} x(t) &= x(0)\,\mathrm{e}^{\lambda t} \\ &= x(0)\,\mathrm{e}^{\lambda_{\mathrm{R}} t}\,\mathrm{e}^{i\,\lambda_{\mathrm{I}} t} \\ &= x(0)\,\mathrm{e}^{\lambda_{\mathrm{R}} t}\,(\cos\lambda_{\mathrm{I}} t + i\sin\lambda_{\mathrm{I}} t). \end{aligned}$$

This solution is bounded for all times $t$ if $\lambda_{\mathrm{R}} \leq 0$ and $\lambda_{\mathrm{I}} \in \Re$. If $\lambda$ is purely real, $x(t)$ merely decays to 0 as $t$ becomes large. If $\lambda$ is purely imaginary $x(t)$ shows oscillatory behavior as a function of $t$.

#### 4.3.1.1 Explicit Euler

The explicit Euler scheme for Eq. 4.29 gives

$$x(t_{n+1}) = x(t_n) + \delta t\, \lambda\, x(t_n).$$

In terms of the $x(t_0) = x(0)$, $x(t_{n+1})$ is given by

$$\begin{aligned} x(t_{n+1}) &= (1 + \lambda\,\delta t)^{n+1} x(0), \\ &= \rho^{n+1} x(0), \end{aligned} \tag{4.30}$$

where $\rho = (1 + \lambda\,\delta t)$ is referred to as the *amplification factor* of the numerical scheme. For the explicit Euler scheme to be bounded and therefore stable at all times (however large is time), we require

$$|\rho| \leq 1. \tag{4.31}$$

That is,

$$\begin{aligned} & |(1 + \lambda\,\delta t)| && \leq && 1, \\ \textit{i.e.,} \quad & |[1 + (\lambda_{\mathrm{R}} + i\,\lambda_{\mathrm{I}})\,\delta t]| && \leq && 1, \\ \textit{i.e.,} \quad & \sqrt{(1 + \lambda_{\mathrm{R}}\delta t)^2 + (\lambda_{\mathrm{I}}\delta t)^2} && \leq && 1, \\ \textit{i.e.,} \quad & (1 + \lambda_{\mathrm{R}}\delta t)^2 + (\lambda_{\mathrm{I}}\delta t)^2 && \leq && 1. \end{aligned} \tag{4.32}$$

not clear!

This inequality defines the *region of stability* of explicit Euler discretization scheme (see Fig. 4.2). Since $\lambda_{\mathrm{R}} < 0$ for our model Eq. 4.29 to be bounded, we can write $\lambda_{\mathrm{R}} = -|\lambda_{\mathrm{R}}|$. Making this substitution in Eq. 4.32

$$\begin{aligned} (1 - |\lambda_{\mathrm{R}}|\,\delta t)^2 + (\lambda_{\mathrm{I}}\delta t)^2 \quad &\leq \quad 1 \\ \textit{i.e.,} \qquad \delta t \leq \frac{2\,|\lambda_{\mathrm{R}}|}{|\lambda_{\mathrm{R}}|^2 + \lambda_{\mathrm{I}}^2}&. \end{aligned} \tag{4.33}$$

Therefore, for the explicit Euler scheme of Eq. 4.29 to be numerically stable, $\delta t$ must fulfil Eq. 4.33. We therefore call the explicit Euler scheme *conditionally stable*, where the condition for stability is given by Eq. 4.33.
If $\lambda$ is purely real (i.e., $\lambda_{\mathrm{I}} = 0$), then

$$\delta t \leq \frac{2}{|\lambda_{\mathrm{R}}|}. \tag{4.34}$$

The explicit Euler scheme is therefore *conditionally stable* when $\lambda$ is purely real, where the condition for stability is given by Eq. 4.34.
If $\lambda$ is purely imaginary (i.e., $\lambda_{\mathrm{R}} = 0$), then

$$\delta t \leq 0. \tag{4.35}$$

This condition, however, cannot be satisfied since $\delta t$ must be greater than 0 by definition. Therefore, the explicit Euler scheme is *unconditionally unstable* when $\lambda$ is purely imaginary. In other words, explicit Euler scheme is unconditionally unstable for purely oscillatory dynamics.

#### 4.3.1.2 Leapfrog

The leapfrog scheme for Eq. 4.29 leads to

$$x(t_{n+1}) = x(t_{n-1}) + 2\delta t\,\lambda\,x(t_n). \tag{4.36}$$

Using the definition of amplification factor $\rho$, we substitute $x(t_{n+1})$ with $\rho^2 x(t_n)$, and $x(t_n)$ with $\rho x(t_{n-1})$ in Eq. 4.36. This substitution results in the following equation:

$$\rho^2 - 2\delta t\lambda\rho - 1 = 0.$$

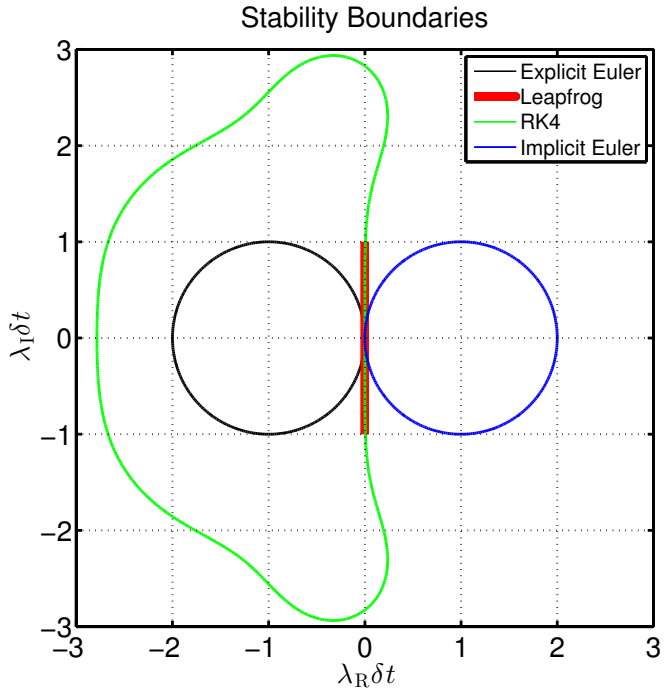



Figure 4.2: The numerical stability boundaries for explicit Euler, leapfrog, Runge-Kutta 4, and Implicit Euler The region of stability for explicit Euler, leapfrog and Runge-Kutta 4 is within the boundary, whereas that of Implicit Euler is outside its boundary.

The two solutions of the above equation are

$$\rho_{1,2} = \lambda\delta t \pm \sqrt{\lambda^2\delta t^2 + 1}.$$

$x(t_{n+1})$ can be written as linear combination of $\rho_1$ and $\rho_2$ so that

$$x(t_{n+1}) = \rho_1 x(t_n) + \rho_2 x(t_n).$$

Therefore, we require

$$\rho_1 \leq 1 \quad \text{and} \quad \rho_2 \leq 1.$$

We find the condition on $\delta t$ that fulfils the above requirement as follows. The product of $\rho_1$ and $\rho_2$, i.e., $\rho_1\rho_2$ = -1. Therefore, $|\rho_1||\rho_2| = 1$. As a consequence, if $\rho_1 > 1$, then $\rho_2 < 1$ and vice versa. Therefore, the only possibility of fulfilling the stability criteria is when $|\rho_1| = |\rho_2| = 1$. Let us find the condition when $|\rho_1| = |\rho_2| = 1$. That is,

$$|\rho_1| = |\lambda\delta t + \sqrt{\lambda^2\delta t^2 + 1}| = 1 \quad \text{and} \quad |\rho_2| = |\lambda\delta t - \sqrt{\lambda^2\delta t^2 + 1}| = 1.$$

Setting $\lambda = -|\lambda_{\mathrm{R}}| + i\lambda_{\mathrm{I}}$, we find that $|\rho_1| = |\rho_2| = 1$ only when $\lambda_{\mathrm{R}} = 0$ and $|\lambda_{\mathrm{I}}|\,\delta t \leq 1$.
Therefore, the leapfrog scheme is *conditionally stable* only for purely oscillatory dynamics.

# 5

# The effect of intrinsic noise on chemical kinetics

*In this chapter:*

- The effect of noise on linear reaction networks
- The effect of noise on nonlinear reaction networks
- The effect of noise on oscillatory reaction networks
- The effect of noise on bi-stable reaction networks

*Learning goals:*

- Know how low copy number and intrinsic noise affect chemical kinetics in linear, nonlinear, oscillatory, and multi-stable reaction networks
- Know under what conditions the stochastic solution converges toward the deterministic solution
- Know when reaction-rate equations are an appropriate tool for modeling kinetics

When the population $\boldsymbol{n}$ increases proportionally with $\Omega$ such that the total mass density of species is constant, decreasing $\Omega$ leads to an increase in intrinsic noise. Intrinsic noise does not only play the role of adding uncertainty to the various statistical estimates of $\boldsymbol{n}$, but it also has been shown to alter chemical kinetics in a non-trivial way [11, 64, 8, 3, 65, 66, 67, 68, 69, 70, 60, 62, **?**, 71]. As a consequence, intrinsic noise may lead to a probability function $P(\boldsymbol{\phi}, t)$ whose mean is not the same as the concentration $\boldsymbol{\phi}$ predicted by the RRE, but they can be qualitatively different. We illustrate the effect of intrinsic noise on different types of nonequilibrium reaction networks:

1. Monostable reaction networks: linear and nonlinear,
2. An oscillatory reaction network (exhibiting limit-cycle oscillations),

3. A multistable reaction network.

These three different types of reaction networks should be sufficient to understand the qualitative effect of intrinsic noise since they span a large spectrum of dynamics exhibited by chemical reaction networks.

## 5.1 Monostable, linear reaction networks

Linear reaction networks are defined as those in which the sum of the stoichiometries of all reactants is less than 2 in every reaction. Consider the following reaction network as an example of a linear reaction network occurring in a reactor of volume $\Omega$:

$$\begin{array}{rcl} \emptyset & \xrightarrow{k_1} & \mathrm{S}_1 \\ \mathrm{S}_1 & \xrightarrow{k_2} & \emptyset. \end{array} \tag{5.1}$$

The RRE for this reaction network is

$$\frac{\mathrm{d}\phi_1}{\mathrm{d}t} = k_1 - k_2\phi_1, \tag{5.2}$$

where $\phi_1$ is the concentration of species $\mathrm{S}_1$. The fixed point of the RRE is $\phi_{1,\mathrm{ss}} = \frac{k_1}{k_2}$. This sole fixed point is stable irrespective of the values of $k_1$ and $k_2$, the system is monostable. This is easily seen: assume that $\phi_1$ is just slightly larger than $\phi_{1,\mathrm{ss}}$, which means that $\phi_1 > \frac{k_1}{k_2}$, hence $\phi_1 k_2 > k_1$ since $k_2$ is non-negative. Then from Eq. 5.2, $\frac{\mathrm{d}\phi_1}{\mathrm{d}t} < 0$, bringing the system back to the steady state. Likewise, if $\phi_1$ is slightly smaller than $\phi_{1,\mathrm{ss}}$, then $\frac{\mathrm{d}\phi_1}{\mathrm{d}t} > 0$, bringing the system back to the steady state. Perturbation in either direction hence decay and the steady state is stable.
The Chapman-Kolmogorov equation of this reaction network is:

$$P(n_1, t+\delta t) = k_1\Omega\delta t P(n_1-1,t) + (n_1+1)k_2\delta t P(n_1+1,t) + \left[1-(k_1\Omega + n_1k_2)\right]\delta t P(n_1,t), \tag{5.3}$$

where we have used Eqs. 2.3 and 2.5 to translate the reaction propensities to the macroscopic rate constant and the reactor volume. The last term is again the probability that no reaction happens within the time interval $\delta t$. Note that due to the Markov assumption, only states that can be reached within at most one reaction need to be considered. Subtracting $P(n_1,t)$ on both sides and dividing the whole equation by $\delta t$ gives:

$$\frac{P(n_1,t+\delta t) - P(n_1,t)}{\delta t} = k_1\Omega P(n_1-1,t) + (n_1+1)k_2 P(n_1+1,t) - (k_1\Omega + n_1k_2)P(n_1,t). \tag{5.4}$$

Taking the limit $\delta t \to 0$, we obtain the CME:

$$\frac{\partial P(n_1,t)}{\partial t} = k_1\Omega P(n_1-1,t) + (n_1+1)k_2 P(n_1+1,t) - (k_1\Omega + n_1k_2)P(n_1,t), \tag{5.5}$$

where $n_1$ is the population of species $\mathrm{S}_1$. Describing a linear chemical reaction system, the CME can be solved analytically. The steady-state or stationary probability function is

$$P_{\mathrm{ss}}(n_1) = \frac{\mathrm{e}^{-(k_1\Omega/k_2)}(k_1\Omega/k_2)^{n_1}}{n_1!}. \tag{5.6}$$

Denoting stochastic variables with an asterisk, the steady-state mean concentration becomes

$$\langle \phi_1^* \rangle_{\rm ss} = \frac{\langle n_1 \rangle_{\rm ss}}{\Omega} = \frac{k_1}{k_2}. \tag{5.7}$$

The mean steady-state concentration hence is the same as the steady-state concentration predicted by the deterministic RRE, for all $\Omega$. This is generally true for linear reaction networks.
The steady-state variance of the concentration is

$$(\langle \phi_1^{*2} \rangle - \langle \phi_1^* \rangle^2)_{\rm ss} = \frac{k_1}{k_2 \Omega}. \tag{5.8}$$

This shows that the steady-state concentration variance decrease as $\Omega^{-1}$ with increase in $\Omega$. This is also generally true for all linear reaction systems.
We set the initial concentration of $S_1$ to 10, the rates $k_1 = 5$ and $k_2 = 10$. Using these parameters we report trajectories and the steady-state probability function obtained using SSA. Figure 5.1A shows the time evolution of the concentration of species $S_1$ from a single SSA run for different $\Omega$'s, and from the RRE. It can be seen that the fluctuations increase with decreasing $\Omega$. This is also apparent in Fig. 5.1B showing $P_{\rm ss}(\phi_1^*)$, which is a Kronecker delta for the deterministic RRE and becomes increasing broader for smaller $\Omega$'s. The variance of $P_{\rm ss}(\phi_1^*)$ is shown in Fig. 5.8 for different $\Omega$. As expected from the analytical result (Eq. 5.8), the variance scales with $\Omega^{-1}$.
In summary, for any monostable linear reaction network, the mean of the concentration probability function $P(\phi, t)$ is equal to the deterministic concentration $\phi(t)$ from RRE. The intrinsic noise in the CME provides zero-mean fluctuations about the deterministic concentrations. Further, for monostable linear reaction networks the variance of concentration decrease as $\Omega^{-1}$ with increasing $\Omega$.

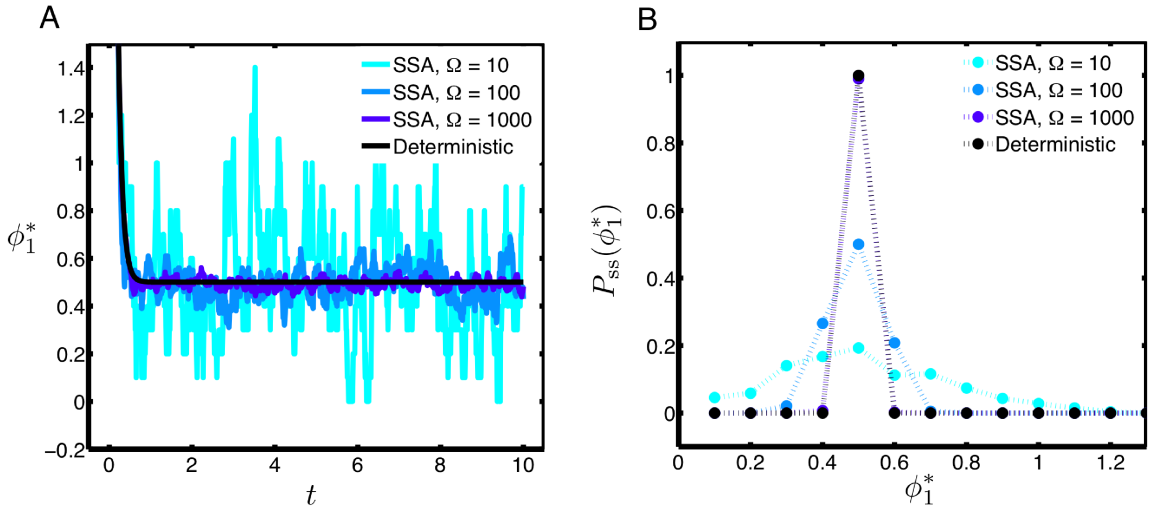



Figure 5.1: (A) Time evolution of the concentration and (B) steady-state concentration probability distribution of species $S_1$ in the monostable, linear reaction network of Eq. 5.1. The results are obtained using the stochastic simulation algorithm (SSA) for different reactor volumes $\Omega$, and using the deterministic reaction rate equation (RRE). The rates used for the simulation are: $k_1 = 5$ and $k_2 = 10$.

## 5.2 Monostable, nonlinear reaction networks

Any reaction network having at least one reaction in which the sum of the stoichiometries of the reactants is greater than 1 is a nonlinear reaction network. As an example consider:

$$\begin{array}{rcl} \emptyset & \xrightarrow{k_1} & S_1 \\ S_1 + S_1 & \xrightarrow{k_2} & \emptyset. \end{array} \tag{5.9}$$

The corresponding RRE is

$$\frac{d\phi_1}{dt} = k_1 - 2k_2\phi_1^2 \tag{5.10}$$

where $\phi_1$ is the concentration of species $S_1$. The fixed point of the RRE is $\phi_{1,ss} = \sqrt{\frac{k_1}{2\,k_2}}$. This fixed point is stable and hence the system is monostable. The stability of the fixed point is again easy to see: Assume that $\phi_1 > \phi_{1,ss}$, hence $\phi_1 > \sqrt{\frac{k_1}{2k_2}}$. Then, $2k_2\phi_1^2 > k_1$, since all quantities are non-negative. This implies that $\frac{d\phi_1}{dt} < 0$, hence bringing the system back to steady state. Likewise, $\phi_1 < \phi_{1,ss}$ implies that $2k_2\phi_1^2 < k_1$ and hence $\frac{d\phi_1}{dt} > 0$, again bringing the system back to steady state.

The CME of this reaction network is

$$\begin{aligned} \frac{\partial P(n_1,t)}{\partial t} &= k_1\Omega P(n_1-1,t) + (n_1+2)(n_1+1)\frac{k_2}{\Omega}P(n_1+2,t) \\ &\quad - \left[k_1\Omega + n_1(n_1-1)\frac{k_2}{\Omega}\right]P(n_1,t), \end{aligned} \tag{5.11}$$

where $n_1$ is the population of species $S_1$. This is one of the very few nonlinear reaction networks for which the steady-state mean concentration (but not the full probability distribution!) can be obtained analytically using moment generating functions [72]. It is given by

$$\langle \phi_1^* \rangle_{\rm ss} = \frac{1}{4\Omega} + \phi_{1,\rm ss} \frac{I_1' \left(4\phi_{1,\rm ss}\Omega\right)}{I_1 \left(4\phi_{1,\rm ss}\Omega\right)}, \tag{5.12}$$

where $I_1(b)$ is the modified Bessel function of the first kind and $I_1'(b) = \frac{{\rm d}I_1(x)}{{\rm d}x}|_{x=b}$. The asymptotic expansion of this expression is:

$$\langle \phi_1^* \rangle_{\rm ss} \quad = \quad \phi_{1,\rm ss} + \frac{\Omega^{-1}}{8} + \frac{3\Omega^{-2}}{128\phi_{1,\rm ss}} + O\left(\Omega^{-3}\right). \tag{5.13}$$

Therefore, we then see that

$$\lim_{\Omega\to\infty} \langle \phi_1^* \rangle_{\rm ss} = \phi_{1,\rm ss}. \tag{5.14}$$

For finite $\Omega$, there is a non-zero difference between the steady-state mean concentration obtained from the CME and the steady-state concentration from the RRE. In general, this difference always persists for nonlinear reaction networks at finite $\Omega$.

Using $k_1 = 5$, $k_2 = 10$ and the initial concentration of $S_1$ set to 10, we see that the steady-state concentration variance scales with $\Omega^{-1}$ as shown in Fig. 5.8. Figure 5.2A shows the the time evolution of the concentration of species $S_1$ obtained from a single SSA run for different $\Omega$'s, and from the RRE. It can be seen that the fluctuations increase with decreasing $\Omega$, evident from the broadening of $P_{\rm ss}(\phi_1^*)$ in Fig. 5.2B in agreement with the scaling of the variance observed in Fig. 5.8. The steady-state mean concentrations are 0.5130, 0.5012 and 0.5001 for $\Omega = 10, 100$ and 1000, respectively. The steady-state concentration $\phi_{1,\rm ss}$ from the RRE is 1/2. This shows that even at a small volume of $\Omega = 10$, the difference between $\phi_{1,\rm ss}$ and $\langle \phi_1^* \rangle_{\rm ss}$ is just about 2.6%. This small difference is specific to the particular system and in general the difference need not so small for other monostable nonlinear reaction networks.

In summary, for any monostable nonlinear reaction network, the mean of the concentration probability function $P(\boldsymbol{\phi}, t)$ is *not* equal to the deterministic concentration $\boldsymbol{\phi}(t)$ obtained from the RRE. The intrinsic noise in the CME deviates the mean concentration from the deterministic concentration. The variance of the concentrations, however, decrease as $\Omega^{-1}$ with increasing $\Omega$. This results in the mean concentration of $P(\boldsymbol{\phi}, t)$ approaching the deterministic concentration with the difference becoming increasingly smaller at larger $\Omega$. For monostable nonlinear reaction networks with multiple species intrinsic noise in the CME can even lead to qualitative difference in the steady-state of the system [73]. Specifically, intrinsic noise can lead to an ordering of steady states that is different from the RRE prediction. This is called "stochastic inversion effect" [73] and renders RRE qualitatively invalid when it occurs.

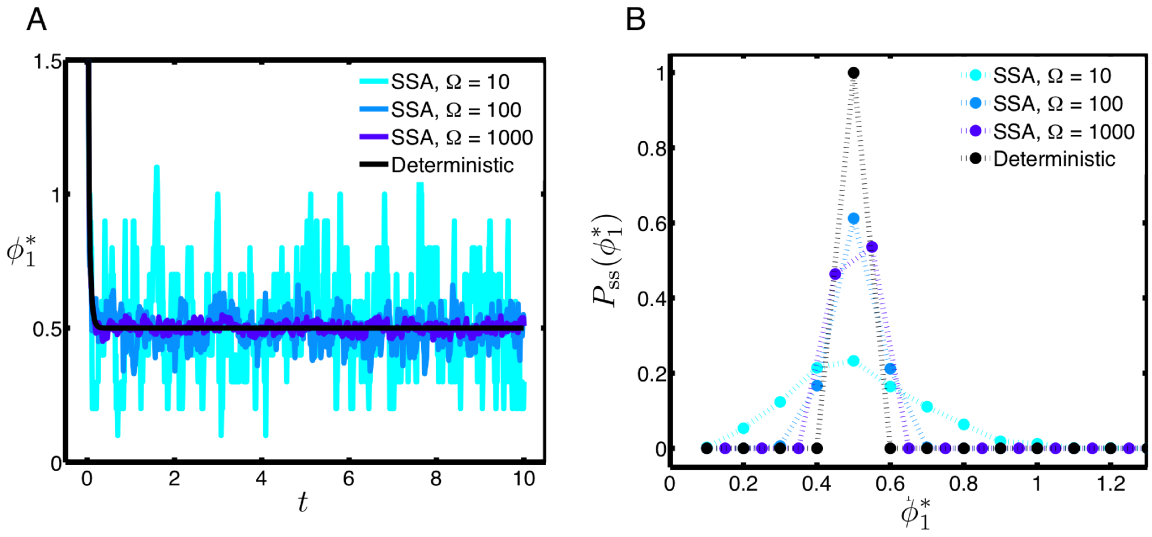



Figure 5.2: (A) Time evolution of the concentration and (B) steady-state concentration probability distribution of species $S_1$ in the monostable, nonlinear reaction network of Eq. 5.9. The results are obtained using the stochastic simulation algorithm (SSA) for different reactor volumes $\Omega$, and using the deterministic reaction rate equation (RRE). The rates used for the simulation are: $k_1 = 5$ and $k_2 = 10$.

## 5.3 Brusselator: an oscillatory reaction network

The Brusselator [74, 75] is a widely used model system for studying oscillatory reaction networks. It is a model system for autocatalytic reactions. Examples of autocatalytic reaction systems include the Belousov-Zhabotinsky reaction and autophosphorylation reactions ubiquitous in biology. Oscillatory reaction networks are also used to model biological rhythms, such as the circadian clock [69, 76, 77, 78, 79] and the glycolytic cycle [75, 80]. The Brusselator reaction network involves two species and is given by

$$\begin{aligned} \emptyset &\xrightarrow{k_1} S_1 \\ S_1 &\xrightarrow{k_2} \emptyset \\ S_1 &\xrightarrow{k_3} S_2 \\ 2S_1 + S_2 &\xrightarrow{k_4} 3S_1. \end{aligned} \tag{5.15}$$

The corresponding RRE is

$$\begin{aligned} \frac{d\phi_1}{dt} &= k_1 - k_2\phi_1 - k_3\phi_1 + k_4\phi_1^2\phi_2 \\ \frac{d\phi_2}{dt} &= k_3\phi_1 - k_4\phi_1^2\phi_2, \end{aligned} \tag{5.16}$$

where $\phi_1$ and $\phi_2$ are the concentrations of $S_1$ and $S_2$, respectively. For simplicity, we set $k_2 = 1$ and $k_4 = 1$. Under these conditions, the fixed point of the deterministic RRE (Eq. 5.16) is given by

$\phi_{1,\mathrm{ss}} = k_1$ and $\phi_{2,\mathrm{ss}} = k_3/k_1$. This fixed point is stable if $k_3 < k_1^2 + 1$, undergoes a Hopf bifurcation at $k_3 = k_1^2 + 1$ and is unstable if $k_3 \geq k_1^2 + 1$. In the latter case, $\phi_1(t)$ and $\phi_2(t)$ exhibit limit-cycle oscillations.
The CME of the reaction network is

$$\begin{aligned}\frac{\partial P(n_1,n_2,t)}{\partial t} &= k_1\Omega P(n_1-1,n_2,t) + (n_1+1)k_2 P(n_1+1,n_2,t)\\ &\quad +(n_1+1)k_3 P(n_1+1,n_2-1,t)\\ &\quad +(n_1-1)(n_1-2)(n_2+1)\frac{k_4}{\Omega^2}P(n_1-1,n_2+1,t)\\ &\quad -\left[k_1\Omega + n_1k_2 + n_1k_3 + n_1(n_1-1)n_2\frac{k_4}{\Omega^2}\right]P(n_1,n_2,t), \qquad (5.17)\end{aligned}$$

where $n_1$ and $n_2$ are the populations of species $S_1$ and $S_2$, respectively. This CME cannot be solved analytically and hence we rely on SSA simulations.
For the simulation, we set $k_1 = 1$ and $k_3 = 1$. For these parameters the fixed point of the RRE is stable and the deterministic solution does not exhibit limit-cycle oscillations. We set the initial concentration of the system to its fixed point, i.e., the initial concentrations of both $S_1$ and $S_2$ are set to 1. As expected Fig. 5.3, shows that the deterministic RRE trajectory stays at the fixed point. As $\Omega$ is decreased, the SSA simulations display increasing fluctuations. Already at a volume of $\Omega = 1000$ the SSA trajectory shows oscillations, even though the deterministic RRE doesn't show any oscillatory behavior. This is evident from the closed-loop phase-space ($\phi_2^*(t)$ versus $\phi_1^*(t)$) trajectory shown in Fig. 5.4.
Figure 5.5 shows the bivariate steady-state probability function $P_{\mathrm{ss}}(\phi_1^*, \phi_2^*)$ for $\Omega = 10$, $\Omega = 100$ and $\Omega = 1000$. As expected, the probability function broadens with decreasing $\Omega$. The steady-state concentration variance of species $S_1$ scales with $\Omega^{-1}$ (see Fig. 5.8). When the fixed point is unstable ($k_1 = 1$, $k_3 = 4$), however, this scaling doesn't hold (See Fig. 5.8) and the variance decreases only slowly with increasing $\Omega$ (variance $\sim \Omega^{-\alpha}$ where $\alpha \ll 1$. $\alpha \approx 0.08$ in the present case).
The emergence of oscillations due to intrinsic noise can be understood as an interplay between the drift term and the diffusion term in the CLE (Eq. 3.20), which can make the particle undergo a circular motion analogous to that of the limit cycle in the deterministic case. As the magnitude of intrinsic noise decreases, the noisy diffusion term becomes weaker until, at very large $\Omega$, the motion is dominated by the deterministic drift (Liouville process, see Eq. 4.9). This interplay between drift and noise have been studied in chemical reaction systems and it has been found to be the reason for oscillations in the stochastic trajectory [62, 81].
In summary, intrinsic noise can lead to oscillations even when the corresponding RRE does not exhibit any. The scaling of the variance of the concentration with $\Omega$ depends on the stability of the fixed point of the RRE. If the fixed point is stable, such that the RRE does not exhibit oscillations, the variance decrease as $\Omega^{-1}$ with increasing $\Omega$. If the fixed point is unstable, such that the RRE exhibits limit-cycle oscillations, the variance decreases much slower with increasing $\Omega$, since the oscillations dominate the fluctuations.
An interesting observation is that while the deterministic model predicts a sudden bifurcation from the non-oscillatory to the oscillatory regime, the onset of oscillations is gradual in the stochastic descriptions. This means that stochastic oscillators are more robust than deterministic ones, where the bifurcation can kill the oscillations. Stochastic oscillators are hence often found in biological systems, where it is important that variations in the rate constants do not suddenly kill the oscillation.

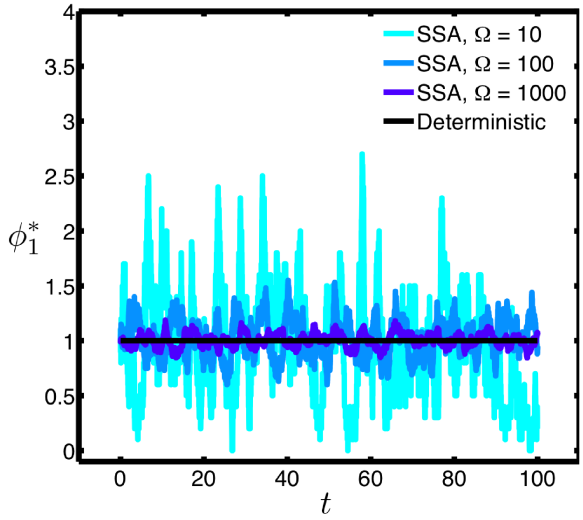



Figure 5.3: Time evolution of the concentration of species $S_1$ in the Brusselator reaction network of Eq. 5.15. The results are obtained using the stochastic simulation algorithm (SSA) for different reactor volumes $\Omega$, and using the deterministic reaction rate equation (RRE). The rates used for the simulation are: $k_1 = 1$, $k_2 = 1$, $k_3 = 1$ and $k_4 = 1$. The initial concentrations of species $S_1$ and $S_2$ are set to 1, the fixed point of the RRE.

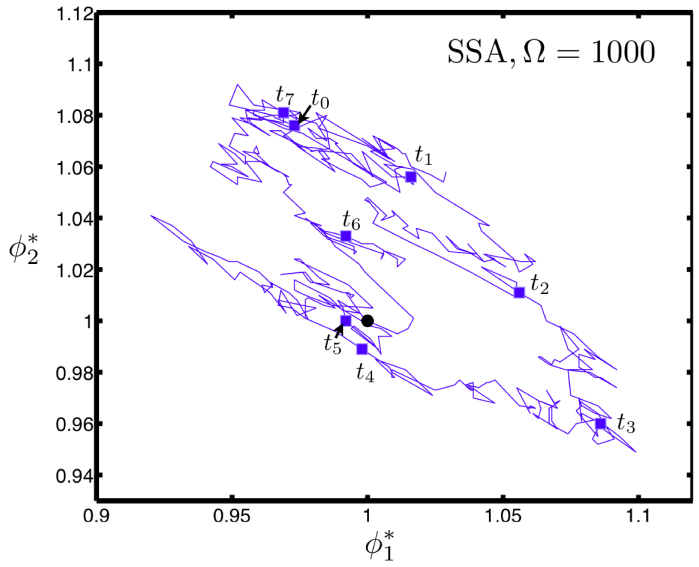



Figure 5.4: Steady-state concentration trajectory in the phase space of the Brusselator reaction network of Eq. 5.15, obtained using the stochastic simulation algorithm (SSA) with a reactor volume of $\Omega = 1000$. The rates used for the simulation are: $k_1 = 1$, $k_2 = 1$, $k_3 = 1$ and $k_4 = 1$. The initial concentrations of species $S_1$ and $S_2$ are set to 1, the fixed point of the RRE (circle). The squares mark the implicit time of the stochastic trajectory in phase space such that $t_{i+1} > t_i$. The time stamps help visualize a clockwise oscillatory behavior.

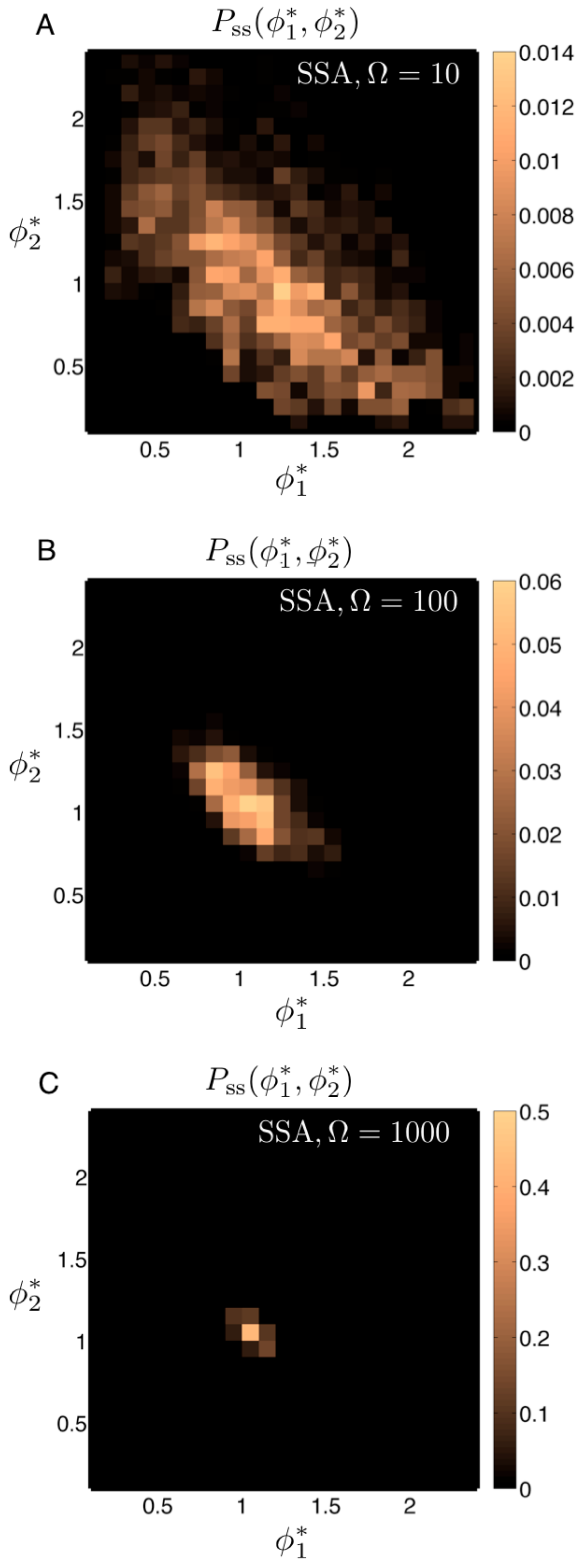



Figure 5.5: Steady-state bivariate probability function of the concetrations of species $S_1$ and $S_2$ in the Brusselator reaction network of Eq. 5.15. The results are obtained using the stochastic simulation algorithm (SSA) for different reactor volumes $\Omega$. The rates used for the simulation are: $k_1 = 1$, $k_2 = 1$, $k_3 = 1$ and $k_4 = 1$. The initial concentration of species $S_1$ and $S_2$ is set to 1, the fixed point of the RRE.

## 5.4 Schlogl model: a bistable reaction network

The Schlogl model [82] is a popular model system for multistable reaction networks. Multistability has also been observed in several biological systems, including the bacterial phenotypic expression [83], lactose utilization network [84], the cell cycle regulation network [85] and in synthetic genetic switches [86]. The Schlogl model is given by

$$\begin{array}{ccc} 2\mathrm{S}_1 & \xrightarrow{k_1} & 3\mathrm{S}_1 \\ 3\mathrm{S}_1 & \xrightarrow{k_2} & 2\mathrm{S}_1 \\ \emptyset & \xrightarrow{k_3} & \mathrm{S}_1 \\ \mathrm{S}_1 & \xrightarrow{k_4} & \emptyset \end{array} \tag{5.18}$$

and it has the RRE

$$\frac{\mathrm{d}\phi_1}{\mathrm{d}t} = k_1\phi_1^2 - k_2\phi_1^3 + k_3 - k_4\phi_1, \tag{5.19}$$

where $\phi_1$ is the concentration of species $\mathrm{S}_1$. The rates are set to $k_1 = 18$, $k_1 = 2.5$, $k_3 = 22$ and $k_4 = 37.5$. For this parameter set, the RRE has 3 fixed points $\phi_{1,\mathrm{ss}} = 1$, $\phi_{1,\mathrm{ss}} = 4$ and $\phi_{1,\mathrm{ss}} = 2.2$. The first two are stable, the third is unstable.
The CME of this system is

$$\begin{aligned} \frac{\partial P(n_1,t)}{\partial t} &= (n_1-1)(n_1-2)\frac{k_1}{\Omega}\Omega P(n_1-1,t) + (n_1+1)n_1(n_1-1)\frac{k_2}{\Omega^2}P(n_1+1,t) \\ &\quad + k_3\Omega P(n_1-1,t) + (n_1+1)k_4 P(n_1+1,t) \\ &\quad - \Bigg[ n_1(n_1-1)\frac{k_1}{\Omega} \\ &\quad + n_1(n_1-1)(n_1-2)\frac{k_2}{\Omega^2} + k_3\Omega + n_1k_4 \Bigg] P(n_1,t) \end{aligned} \tag{5.20}$$

and analytically intractable. We set the initial concentration of species $\mathrm{S}_1$ to 1, which is one of the stable fixed points of the RRE.
Figure 5.6 shows that in the deterministic case the concentration of species $\mathrm{S}_1$ remains unchanged from the initial concentration at the stable fixed point of the system. Using SSA, we see that for $\Omega = 1000$ the stochastic trajectory shows small fluctuations around the deterministic RRE (Fig. 5.6). This can also be seen from the steady-state probability function $P_{\mathrm{ss}}(\phi_1^*)$ (Fig. 5.7C) which is a symmetric, narrow and unimodal distribution with its mean very close to the stable fixed point $\phi_{1,\mathrm{ss}} = 1$ of the RRE. For $\Omega = 100$ the stochastic trajectories switch (jump) back and forth between the neighborhoods of the two stable fixed points (Fig. 5.6). This can also be seen in the steady-state probability function $P_{\mathrm{ss}}(\phi_1^*)$ (Fig. 5.7B) becoming bimodal with maxima close to the two stable fixed points ($\phi_{1,\mathrm{ss}} = 1$, $\phi_{1,\mathrm{ss}} = 4$) of the RRE, and a minimum close to the unstable fixed point $\phi_{1,\mathrm{ss}} = 2.2$ of the RRE. At $\Omega = 10$, the fluctuations of the stochastic trajectory become even larger (Fig. 5.6). The corresponding steady-state probability function $P_{\mathrm{ss}}(\phi_1^*)$ (Fig. 5.7A) is again unimodal, but asymmetric with a long tail. Because of these changes in the qualitative properties of the steady-state probability function, the scaling of the steady-state concentration variance falls in two regimes (Fig. 5.8). At low intrinsic noise or large $\Omega$ when $P_{\mathrm{ss}}(\phi_1^*)$ is symmetric, the variance scales with $\Omega^{-1}$. At large intrinsic noise or small $\Omega$ when $P_{\mathrm{ss}}(\phi_1^*)$ is asymmetric or bimodal, the

variance scales with $\Omega^{-\alpha}$ where $\alpha \ll 1$. In the present case, we estimated this exponent $\alpha$ to be approximately 0.12. In addition, the large $\Omega$ regime is $\Omega \geq 500$ and small $\Omega$ regime is $\Omega \leq 400$ in the present case. The sudden jump observed in the scaling of the variance from the large $\Omega$ regime to the small $\Omega$ regime is brought about by the change in $P_{\rm ss}(\phi_1^*)$ from being a sharp unimodal distribution at $\Omega = 500$ to a bimodal distribution at $\Omega = 400$. This change in $P_{\rm ss}(\phi_1^*)$ is effected by the fluctuations becoming large enough that the concentration reaches the neighborhood of the unstable fixed point, from where the concentration is then probabilistically shuttled to the neighborhood of the other stable fixed point.

In summary, intrinsic noise in a multistable reaction network can lead to switching behavior between the neighborhoods of the multiple stable fixed points. Due to the associated qualitative change in the steady-state concentration probability function, the variance scales as $\Omega^{-\alpha}$ with $\alpha \ll 1$ for small $\Omega$. For large enough $\Omega$ the variance scales with $\Omega^{-1}$.

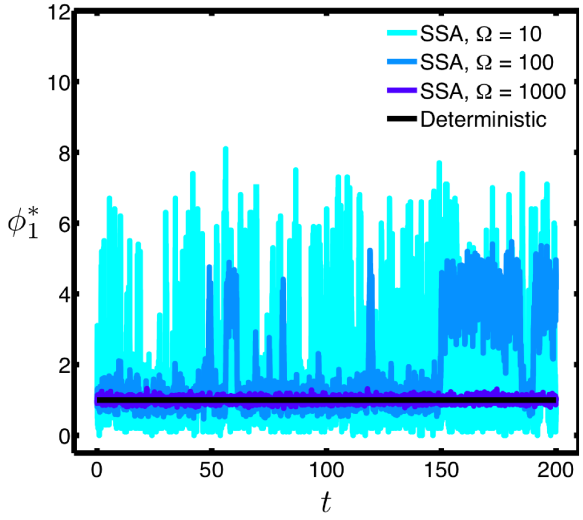



Figure 5.6: Time evolution of the concentration of species $S_1$ in the Schlogl model of Eq. 5.18. The results are obtained using the stochastic simulation algorithm (SSA) for different reactor volumes $\Omega$, and using the deterministic reaction rate equation (RRE). The rates used for the simulation are: $k_1 = 18$, $k_1 = 2.5$, $k_3 = 22$ and $k_4 = 37.5$. The initial concentration of species $S_1$ is set to 1, which is one of the stable fixed points of the RRE.

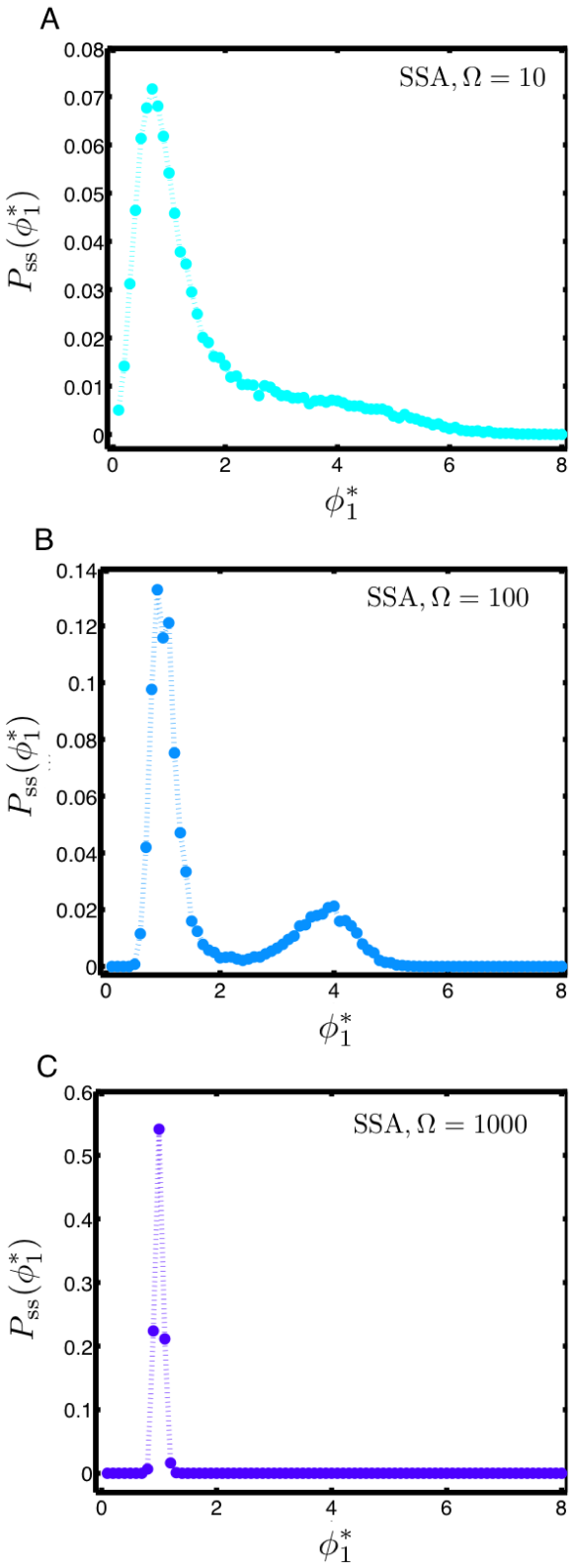



Figure 5.7: Steady-state probability function for the concentration of species $S_1$ in the Schlogl model of Eq. 5.18. The results are obtained using the stochastic simulation algorithm (SSA) for different reactor volumes $\Omega$, and using the deterministic reaction rate equation (RRE). The rates used for the simulation are: $k_1 = 18$, $k_1 = 2.5$, $k_3 = 22$ and $k_4 = 37.5$. The initial concentration of species $S_1$ is set to 1, one of the stable fixed points of the RRE.

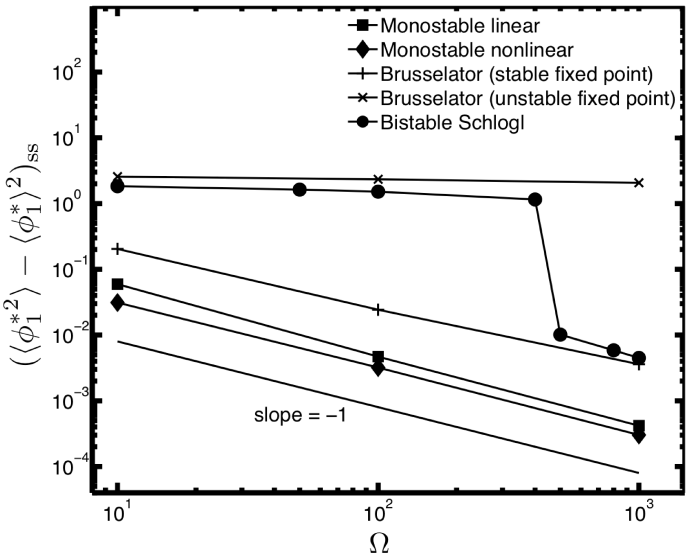



Figure 5.8: Scaling of the steady-state concentration variance of species $S_1$ with the reactor volume $\Omega$ for four different types of reaction networks: monostable linear (Sec. 5.1), monostable nonlinear (Sec. 5.2), non-oscillatory Brusselator, oscillatory Brusselator (Sec. 5.3) and the Bistable Schlogl (Sec. 5.4) models. All results are obtained using the stochastic simulation algorithm (SSA). The parameters used for the simulations are given in the corresponding sections that introduce these reaction networks.

# 6

# Reaction networks with time delays

*In this chapter:*

- Chemical reactions with time delays
- Priority queues to efficiently manage pending reactions
- Piecewise constant time-varying propensities
- Consuming and non-consuming chemical reactions
- Extending partial propensities to networks with time delays

*Learning goals:*

- Be able to derive the waiting time distribution for time-varying propensities
- Be able to implement priority queues with $\log_2 \Delta$ insert complexity
- Understand how partial propensities sample the next reaction
- Be able to use the inversion method for a non-analytic piecewise constant cumulative distribution
- Be able to derive or explain the algorithmic complexity of dDM

All models and simulation methods presented so far assume instantaneous execution of reactions. The population of species is hence instantaneously updated at the time of reaction firing. In many systems, such as gene expression networks in biological cells, the initiation of the reaction and the formation of the products, however, is not instantaneous. In gene expression, the initiation of a reaction corresponds to the binding of the transcription factor to the gene. The product, the ribonucleic acid (RNA), is only formed once the RNA-polymerase finished scanning the entire

gene. There is hence a time delay between the initiation of the reaction and formation of the products. The average transcription and translation speeds in eukaryotic cells, for example, are 20 nucleotides per second and 2 codons per second, respectively [87, 88, 89]. This amounts to a coarse-grained modeling of the fundamental processes that mediate the delay, lumping them into a single delay time. Nevertheless, such approaches are valuable and have been used, for instance, to implicate delay along with intrinsic noise in chemical reactions to tune or induce circadian-rhythmic oscillations in drosophila [90, 91], oscillations in other biomolecular clocks such as in the dynamics of messenger-RNA for Notch signaling molecules [92] and oscillations in gene regulatory networks [88]. Famous oscillatory systems, like the Repressilator, only oscillate if there are time delays in the reactions. Without delays, there are no oscillations. Also, the magnitude of the delays tunes the oscillation frequency.

Delays in chemical reactions render the kinetics non-Markovian. This is because the next population state of the system depends on the current population as well as the population of the system at previous times when still unfinished reactions have been initiated. This non-Markovian process can be accounted for by the delay CME [88, 92, 93]. The delay CME (dCME) is analogous to the classical CME and in the limit of infinite reactor volume $\Omega$ (or infinite population) it tends to the corresponding delay RRE (dRRE), which is a delay differential equation [88, 92, 93]. In order to simulate stochastic chemical kinetics of reaction networks with delays, the delay stochastic simulation algorithm (dSSA) is available [88, 92, 89, 93], extending SSAs to properly account for the effects of non-zero reaction durations. The dSSA samples exact trajectories from the dCME [92, 89, 93]. Accelerated approximate methods to sample from the dCME are also available [94, 95]. The main difference between the dSSA and the conventional SSA is that the reaction propensities (i.e., the probability rate of a reaction firing) may change in the time *between* two reaction initiations (firings), as a result of pending reactions finishing meanwhile. This renders the random variable for the time to the next reaction $\tau$ and the index of the next reaction $\mu$ mutually dependent. In contrast to conventional SSAs, where $\tau$ and $\mu$ are independent random variables (see Eq. 2.27 in Sec. 2.2).

## 6.1 The delay stochastic simulation algorithm (dSSA)

Consider a network of $M$ chemical reactions among $N$ species. Assume that a subset of these $M$ reactions incur a delay. If a reaction involves no delay (hereafter denoted as $\mathrm{R_{D0}}$), it completes instantaneously and the populations of reactants and products are immediately updated. If a reaction incurs a delay, its products are formed only after a delay $d_\mu$ from reaction initiation. We classify delay reactions depending on when the reactants are consumed into *non-consuming* (denoted $\mathrm{R_{D1}}$) and *consuming* (denoted $\mathrm{R_{D2}}$) ones [89]. In non-consuming delay reactions, the population of reactants is only updated once the products have formed, thus after the delay $d_\mu$. In consuming delay reactions, the population of the reactants is updated immediately upon reaction initiation, but the products only form after the delay $d_\mu$. In the following, we measure time globally, i.e. relative to time $t = 0$. This is in contrast to the local (relative to the current time $t$) times used in conventional SSA formulations. We denote the global time of firing (initiation) of the next reaction as $\tau^{\mathrm{g}} = t + \tau$ and the global time at which the products of a delay reaction $\mu$ are formed as $d_\mu^{\mathrm{g}} = t + \tau + d_\mu$.

Assume that at some time $t$ there are $\Delta$ pending (ongoing) delay reactions that will finish at later global times $T_1^{\mathrm{g}}, T_2^{\mathrm{g}}, T_3^{\mathrm{g}}, \ldots, T_\Delta^{\mathrm{g}}$. We assume that the list of pending reactions is ordered according to ascending global completion times, thus $T_i^{\mathrm{g}} \leq T_{i+1}^{\mathrm{g}}$ $i = 1, \ldots, \Delta - 1$ (e.g., in a priority queue data

structure). Furthermore, we define $T_0^{\mathrm{g}} = t$ and $T_{\Delta+1}^{\mathrm{g}} = \infty$. As in classical SSA, the time to the next reaction $\tau$ (or the global time of firing of the next reaction, $\tau^{\mathrm{g}}$) and the index of the next reaction $\mu$ are sampled in order to propagate the system from reaction event to reaction event. In classical SSA, all reactions complete instantaneously, i.e., reaction initiation and completion happen at the same time. Therefore, the reaction propensities remain unchanged during the time interval $[t, t+\tau)$. This, however, is not the case in delay SSAs, where the reaction propensities change whenever a pending reaction completes. The renders the propensity a piecewise constant function that jumps at each reaction completion event. Accounting for these inter-firing changes of the propensities, the probability distribution functions for the global time of firing (initiation) of the next reaction $f_\tau(\tau^{\mathrm{g}})$ and of the index of the reaction $f_\mu(\mu)$ are given by: [89]

$$\begin{aligned} f_\tau(\tau^{\mathrm{g}}) &= a(T_i^{\mathrm{g}}) \exp\left( -\sum_{j=0}^{i-1} a(T_j^{\mathrm{g}})(T_{j+1}^{\mathrm{g}} - T_j^{\mathrm{g}}) - a(T_i^{\mathrm{g}})(\tau^{\mathrm{g}} - T_i^{\mathrm{g}}) \right), \\ \tau^{\mathrm{g}} &\in [T_i^{\mathrm{g}}, T_{i+1}^{\mathrm{g}}), \quad i = 0, \ldots, \Delta\,, \end{aligned} \tag{6.1}$$

and

$$f_\mu(\mu) = \frac{a_\mu(T_i^{\mathrm{g}})}{a(T_i^{\mathrm{g}})}, \quad \mu = 1, \ldots, M, \quad \tau^{\mathrm{g}} \in [T_i^{\mathrm{g}}, T_{i+1}^{\mathrm{g}})\,. \tag{6.2}$$

Here, $a_\mu(t)$ is the reaction propensity of reaction $\mu$ at global time $t$ and $a(t)$ is the total propensity of all reactions at global time $t$.
Understanding where these equations come from is the whole key to understanding dSSA. The first equation, the probability distribution for the waiting times, can be understood as follows: in classical non-delay SSA, the distribution of waiting times $\tau$ until any next reaction happens is $a\mathrm{e}^{-a\tau}$, where the total propensity $a$ is a constant. This is a continuous-time Bernoulli process with constant intensity $a$. If $a$ is a function of time, the rate of events has to be integrated up, rendering the waiting time distribution

$$Z \exp\left[ -\int_0^\tau a(t)\mathrm{d}t \right], \tag{6.3}$$

where $a(t)$ is now a continuous function of time and $Z$ the normalization constant (partition function). In the case here, $a(t)$ is piecewise constant, such that the integral can be computed as:

$$\int_0^\tau a(t)\mathrm{d}t = a(T_0^g)(T_1^g - T_0^g) + a(T_1^g)(T_2^g - T_1^g) + \ldots + a(T_i^g)(\tau - T_i^g)\,, \tag{6.4}$$

where $\tau \in [T_i^g, T_{i+1}^g)$. This is exactly the exponent in Eq. 6.1. The partition function $Z$ in this case is also known, as it simply is the total propensity of the reaction network at that time. Once the global time $\tau^g$ of the next reaction is known, Eq. 6.2 samples the index of the next reaction in exactly the same way that DM does, but using the propensity values at that time.

### 6.1.1 The delay direct method (dDM)

In dDM, as presented by Cai et al. [89] and summarized in Table 6.1, the global time of firing of the next reaction $\tau^{\mathrm{g}}$ is obtained from Eq. 6.7 using linear search in order to sample the interval $p$ such that

$$p = \max\left[ i \;:\; r_1 \geq F(T_i^{\mathrm{g}}) \right] \tag{6.5}$$

with $\tau^{\mathrm{g}} \in [T_p^{\mathrm{g}}, T_{p+1}^{\mathrm{g}})$ and $r_1$ a uniform random number in [0,1). The integer $0 \leq p \leq \Delta$ hence indicates the number of pending reactions that finish before the firing of the next reaction. Here, $F(\cdot)$ is the cumulative distribution function of the probability density function $f_\tau(\tau^{\mathrm{g}})$ (Eq. 6.1). It is given by:

$$\begin{aligned} F(\tau^{\mathrm{g}}) &= 1 - \exp\left( -\sum_{j=0}^{i-1} a(T_j^{\mathrm{g}})(T_{j+1}^{\mathrm{g}} - T_j^{\mathrm{g}}) - a(T_i^{\mathrm{g}})(\tau^{\mathrm{g}} - T_i^{\mathrm{g}}) \right), \\ \tau^{\mathrm{g}} &\in [T_i^{\mathrm{g}}, T_{i+1}^{\mathrm{g}}), \quad i = 0, \ldots, \Delta\,. \end{aligned} \tag{6.6}$$

Note that in order to find $p$, we have to keep track of the change in $a$ whenever a pending reaction finishes. This is done by successively updating the population, affected propensities $a_\mu$, and the total propensity $a$ every time a pending reaction completes. Therefore, $p$ is the search depth needed to sample $\tau^{\mathrm{g}}$.
Once the interval $p$ is determined, $\tau^{\mathrm{g}}$ is calculated as

$$\tau^{\mathrm{g}} = T_p^{\mathrm{g}} + \frac{-\log(1 - r_1) - \sum_{j=0}^{p-1} a(T_j^{\mathrm{g}})(T_{j+1}^{\mathrm{g}} - T_j^{\mathrm{g}})}{a(T_p^{\mathrm{g}})}\,, \tag{6.7}$$

such that

$$\tau^{\mathrm{g}} \in [T_p^{\mathrm{g}}, T_{p+1}^{\mathrm{g}})\,.$$

The whole procedure of sampling $\tau^g$ is sampling a random number from Eq. 6.1 using the inversion method. This is done by inverting the cumulative distribution in Eq. 6.6. However, unlike in standard SSA, the inversion cannot be done analytically, because the function jumps whenever a pending reaction finishes. In-between jumps, it has smooth exponential branches. The algorithm hence first performs a linear search in order to determine between which two jumps the $y = r_1$ lies. Then, the exponential branch between the jumps is analytically inverted in Eq. 6.7. Of course, the same $r_1$ is used, since we want to find its single inverse.
The index $\mu$ of the next reaction is also obtained by linear search. Unlike in Gillespie's original direct method (DM), however, the probability distribution function of $\mu$ depends on the interval $p$, where $p$ is an integer such that $0 \leq p \leq \Delta$ (In Gillespie's non-delay DM, $p$ is fixed to 0 since there are no pending reactions). The next reaction is hence always sampled *after* $p$ has been found. Using a uniform random number $r_2 \in [0, 1)$, $\mu$ is found such that

$$\mu = \min\left[ \mu' \;:\; r_2 a(T_p^{\mathrm{g}}) \;<\; \sum_{i=1}^{\mu'} a_i(T_p^{\mathrm{g}}) \right]\,. \tag{6.8}$$

This is identical to the standard SSA, except that the propensity values of the proper time interval have to be used. The algorithm of the delay direct method (dDM) [89] is summarized in Table 6.1. It is built around a list of global completion times of the pending delay reactions, maintained in ascending order (i.e., a priority queue). The computational cost of this algorithm is determined by the following steps:

**Update step:** In a strongly coupled reaction network, firing of one reaction can potentially affect all propensities. Hence, the computational cost of updating the reaction propensities is $O(M)$,

where $M$ is the number of reactions in the network. For a weakly coupled reaction network the update step is $O(1)$ since the number of propensities affected by any reaction is (by definition of a weakly coupled network) bounded by a constant.

**Sampling the global time of the next reaction:** The computational cost of sampling the global time of firing (initiation) of the next reaction, $\tau^{\mathrm{g}}$, is $O(pM)$ for a strongly coupled reaction network. Here, $p$ is the search depth to locate $\tau^{\mathrm{g}}$ according to Eq. 6.5. This is because the number of times the propensities need to be updated due to pending reactions finishing is $p$ when $\tau^{\mathrm{g}} \in [T_p^{\mathrm{g}}, T_{p+1}^{\mathrm{g}})$. In each of these $p$ updates, $O(M)$ propensities need to be updated. Deleting the $p$ pending reactions that finish is $O(p)$. Similarly, for a weakly coupled reaction network, the computational cost of sampling $\tau^{\mathrm{g}}$ is $O(p)$. In C++, we store the list of the global finishing times of pending reactions in the `multiset` standard template library (STL) container and use the provided methods to add and remove pending reactions.

**Sampling the index of the next reaction:** The index of the next reaction is found by linear search across the $M$ propensities. The computational cost of this operation is $O(M)$. If the sampled reaction is a delay reaction, it is added to the list of pending reactions, along with its global completion time. Inserting a new reaction such a way that the global times of completion of pending reactions remain sorted in ascending order is $O(\log_2 \Delta)$, where $\Delta$ is the number of pending reactions currently in the list. This can be done using bisection search, which is equivalent to a binary search tree.

In summary, the computational cost of dDM is $O(pM + M + \log_2 \Delta)$ for strongly coupled reaction networks. This is equivalent to $O(pM + \log_2 \Delta)$ for $p > 0$. For weakly coupled reaction networks, the computational cost is $O(p + M + \log_2 \Delta)$. Note that when there are no delay reactions, and hence no global pending times need to be inserted in the list, the computational cost of dDM is $O(M)$, as for Gillespie's DM.

## 6.2 The delay partial-propensity SSA

We present a partial-propensity formulation of the exact dSSA for chemical reaction networks with delays: the delay partial-propensity direct method (dPDM). Since sampling the index of the next reaction in dDM and DM is identical up to the temporal change of the propensities, we use the same partial-propensity structure and sampling idea. Sampling the time to the next reaction is identical to dDM. The main change is that the priority queue of pending reaction requires a few additional bookkeeping data structures. The computational cost of dPDM is $O(pN + \log_2 \Delta)$ for strongly coupled networks and $O(p + N + \log_2 \Delta)$ for weakly coupled ones. As a result of using partial propensities, the number of reactions $M$ in the computational cost of dDM is replaced by the usually smaller number of species $N$. The dPDM formulation is thus especially efficient when $p$ and $\Delta$, which are network-specific parameters that are independent of the simulation method, do not scale faster than $O(N)$. In addition, the linear dependence of the computational cost on $N$ makes dPDM especially efficient for strongly coupled reaction networks, where $M$ grows much faster than $N$ with network size.

1. Initialization: set $t \leftarrow 0$, $\delta a \leftarrow 0$, and the number of pending reactions $\Delta \leftarrow 0$; initialize the population vector $\boldsymbol{n}$, the propensities $a_\mu$, and the total propensity $a = \sum_\mu a_\mu$.
2. Sample the global time of firing of the next reaction, $\tau^{\mathrm{g}}$: First, perform linear search to find the search depth $p$ such that $p \in [T_p^{\mathrm{g}}, T_{p+1}^{\mathrm{g}})$ according to Eq. 6.5. During this, update the population and the propensities every time a pending reaction finishes, iteratively propagating the system to time $T_p^g$. Then compute $\tau^{\mathrm{g}}$ according to Eq. 6.7. Update $\Delta \leftarrow \Delta - p$ and set $t \leftarrow \tau^{\mathrm{g}}$.
3. Sample the index of the next reaction $\mu$ according to Eq. 6.8 using linear search.
4. If $\mu$ is a delay reaction, insert $t + d_\mu$ into the list that stores the global finishing times of the pending reactions. Use bisection search to insert at the proper position such that the list is maintained in ascending order; increment $\Delta \leftarrow \Delta + 1$.
4. Update $\boldsymbol{n}$ depending on the delay type of reaction $\mu$.
5. Update the affected $a_\mu$'s using a dependency graph and calculate the change in total propensity $\delta a$.
6. Update $a \leftarrow a + \delta a$.
7. Go to step 2.

Table 6.1: Outline of the algorithm for the delay direct method (dDM) with global times. In C++, the list of global finishing times of pending reactions can conveniently be stored in a `multiset` standard template library container.

### 6.2.1 Detailed description

Like in PDM, the partial propensities in dPDM are stored in a partial-propensity structure $\mathbf{\Pi} = \{\mathbf{\Pi}_i\}_{i=0}^{N}$, the reaction indices in look-up table $\mathbf{L} = \{\mathbf{L}_i\}_{i=0}^{N}$, the sum of partial propensity in a group in the array $\mathbf{\Lambda}$ and sum of propensities in a group in the array $\mathbf{\Sigma}$.
After each reaction event (reaction initiation or completion) the population $\boldsymbol{n}$, the partial propensities $\Pi_{i,j}$, the $\Lambda_i$'s, and the $\Sigma_i$'s need to be updated. Which values need to be updated depends on the type of event that happened (firing of a non-delay reaction, initiation of a non-consuming delay reaction, initiation of a consuming delay reaction, or completion of a delay reaction). We efficiently implement the updates using the following data structures:

$\mathbf{U}^{(1)}$: an array of $M$ arrays, where the $i^{\text{th}}$ array contains the indices of all species involved in the $i^{\text{th}}$ reaction.

$\mathbf{U}^{(2)}$: an array of $M$ arrays containing the corresponding stoichiometry (the change in population of each species upon reaction) of the species stored in $\mathbf{U}^{(1)}$.

$\mathbf{U}^{(1)}_{(-)}$: an array of $M$ arrays, where the $i^{\text{th}}$ array contains the indices of all species that are reactants in the $i^{\text{th}}$ reaction.

$\mathbf{U}^{(2)}_{(-)}$: an array of $M$ arrays containing the corresponding stoichiometry of the reactant species stored in $\mathbf{U}^{(1)}_{(-)}$. $\mathbf{U}^{(1)}_{(-)}$ and $\mathbf{U}^{(2)}_{(-)}$ constitute the sparse-representation of the reactant stoichiometry matrix $\boldsymbol{\nu}^{-}$.

$\mathbf{U}^{(1)}_{(+)}$: an array of $M$ arrays, where the $i^{\text{th}}$ array contains the indices of all species that are products in the $i^{\text{th}}$ reaction.

$\mathbf{U}^{(2)}_{(+)}$: an array of $M$ arrays containing the corresponding stoichiometry of the product species stored in $\mathbf{U}^{(1)}_{(+)}$. $\mathbf{U}^{(1)}_{(+)}$ and $\mathbf{U}^{(2)}_{(+)}$ constitute the sparse-representation of the product stoichiometry matrix $\boldsymbol{\nu}^{+}$.

$\mathbf{U}^{(3)}$: an array of $N$ arrays, where the $i^{\text{th}}$ array contains the indices of all entries in $\mathbf{\Pi}$ that depend on $n_i$.

We also maintain a priority queue $\boldsymbol{T}$ that stores the global times ($T_i^{\text{g}},\ i = 0, \ldots, \Delta$) of all $\Delta$ pending reactions in ascending order. The corresponding indices and delay types ($\text{R}_{\text{D}0}$, $\text{R}_{\text{D}1}$, or $\text{R}_{\text{D}2}$) of the reactions are stored in the lists $\boldsymbol{\mu}^{(\text{D})}$ and $\boldsymbol{D}$, respectively.
In dPDM, the global time of firing (initiation) of the next reaction, $\tau^{\text{g}}$, and the index of the next reaction are mutually dependent. First, the interval $p$ is found according to Eq. 6.5 using linear search such that the global time of firing of the next reaction $\tau^{\text{g}} \in [T_p^{\text{g}}, T_{p+1}^{\text{g}})$. This tells us between which two reaction completion events the next firing or initiation event happens (see Fig. 6.1A). The difference between dPDM and dDM in sampling $p$ is the mechanism of updating the total propensity $a(T_i^{\text{g}})$ each time a pending reaction completes and is removed from the queue of pending reactions. In dPDM, we make use of the partial propensities $\mathbf{\Pi}$ and the associated data structures to update $a$. For instance, assume that $\tau^{\text{g}} \in [T_1^{\text{g}}, T_2^{\text{g}})$ and the reaction type associated with the global completion time $T_1^{\text{g}}$ is $\text{R}_{\text{D}2}$ (consuming delay reaction). In this case, we update $\boldsymbol{n}$ using $\mathbf{U}^{(1)}_{(+)}$ and $\mathbf{U}^{(2)}_{(+)}$. If the finishing reaction is of type $\text{R}_{\text{D}1}$ (non-consuming delay reaction), $\boldsymbol{n}$ is updated using $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$. Subsequently, $\mathbf{\Pi}$ and the associated data structures are updated

using $\mathbf{U}^{(3)}$, thereby obtaining $\delta a$ (the change in $a$) and hence the new $a$. All these updates are done at the completion times of each pending reaction until the interval containing the global time of firing (initiation) of the next reaction is reached and all $p$ pending reactions that have completed are removed from the list $\boldsymbol{T}$. Then, the global time of firing (initiation) of the next reaction, $\tau^{\mathrm{g}}$, within that interval is calculated according to Eq. 6.7.

For sampling the index $\mu$ of the next reaction, we use a single uniformly distributed random number $r_2 \in [0,1)$ to (a) sample the group index $I$ using linear search such that

$$I = \min\left[I' \ : \ r_1 a(T_p^{\mathrm{g}}) < \sum_{i=0}^{I'} \Sigma_i(T_p^{\mathrm{g}})\right] \tag{6.9}$$

and (b) sample the element index $J$ in $\mathbf{\Pi}_I$ using linear search such that

$$J = \min\left[J' \ : \ r_1 a(T_p^{\mathrm{g}}) < \sum_{j=1}^{J'} n_I \Pi_{I,j}(T_p^{\mathrm{g}}) + \left(\sum_{i=0}^{I} \Sigma_i(T_p^{\mathrm{g}})\right) - \Sigma_I(T_p^{\mathrm{g}})\right] \tag{6.10}$$

if $\tau^{\mathrm{g}} \in [T_p^{\mathrm{g}}, T_{p+1}^{\mathrm{g}})$ (see Fig. 6.1B). The sampling of $J$ can be performed efficiently as described in Eq. 2.34. The indices $I$ and $J$ are then translated back to the reaction index $\mu$ using the look-up table $\mathbf{L}$, thus $\mu = \mathrm{L}_{I,J}$.

Once the index of the next reaction is sampled, we ascertain the type of the reaction and initiate it. If $\mu$ is a non-delay (type $\mathrm{R_{D0}}$) reaction, the population $\boldsymbol{n}$ is immediately updated using $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$. Subsequently, $\mathbf{\Pi}$ is updated using $\mathbf{U}^{(3)}$. If $\mu$ is a non-consuming delay reaction (type $\mathrm{R_{D1}}$), $\boldsymbol{n}$ and $\mathbf{\Pi}$ are not updated at the time of reaction initiation. Instead, the attributes of this delay reaction (its global time of completion, index, and type) are inserted into $\boldsymbol{T}$, $\boldsymbol{\mu}^{(\mathrm{D})}$, and $\boldsymbol{D}$, respectively. We ensure that the global completion times in $\boldsymbol{T}$ are maintained in ascending order by inserting at the appropriate location, which is found using bisection search. If $\mu$ is a consuming delay reaction (type $\mathrm{R_{D2}}$), $\boldsymbol{n}$ is immediately updated using $\mathbf{U}_{(-)}^{(1)}$ and $\mathbf{U}_{(-)}^{(2)}$. Subsequently, $\mathbf{\Pi}$ is updated using $\mathbf{U}^{(3)}$. In addition, the attributes of this reaction are inserted into $\boldsymbol{T}$, $\boldsymbol{\mu}^{(\mathrm{D})}$, and $\boldsymbol{D}$ at the appropriate location, again found by bisection search.

In summary, dPDM is an exact formulation of dSSA, generalizing PDM to handle reactions with delays according to the probability distribution functions of dSSA (Eqs. 6.1 and 6.2). The detailed algorithm of dPDM is given in Table 6.2. The computational cost of dPDM is $O(pN + \log_2 \Delta)$ for strongly coupled reaction networks and $O(p + N + \log_2 \Delta)$ for weakly coupled ones, as shown in Sec. 6.2.2.

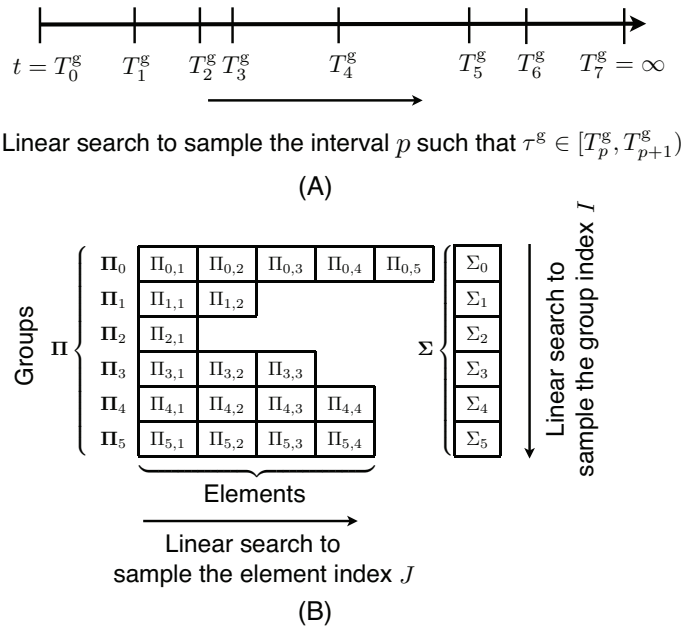



Figure 6.1: Illustration of the main steps in dPDM. (A) Illustration of the linear search to find the interval $p$ such that the global time of firing (initiation) of the next reaction $\tau^g \in [T_p^g, T_{p+1}^g)$. In this figure, the number of pending reactions $\Delta = 6$. (B) Illustration of the partial-propensity structure $\mathbf{\Pi}$ and the grouping based on the index of the common factored-out reactant. The group index $I$ of the next reaction is sampled using linear search over the total propensities of the groups, $\Sigma_i$. The element index $J$ within the selected group is found using linear search over the partial propensities stored in group $I$.

1. Initialization: set $t \leftarrow 0$, $\delta a \leftarrow 0$, and the number of pending reactions $\Delta \leftarrow 0$; initialize the population vector $\boldsymbol{n}$, the partial propensities $\boldsymbol{\Pi}$, the group sum array $\boldsymbol{\Lambda}$, $\boldsymbol{\Sigma}$, and the total propensity $a \leftarrow \sum_{i=0}^{N} \Sigma_i$; initialize $\mathbf{T}$, $\mathbf{D}$, and $\boldsymbol{\mu}^{(\mathrm{D})}$ (these are empty at this stage); initialize the update structures $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(1)}_{(-)}$, $\mathbf{U}^{(2)}_{(-)}$, $\mathbf{U}^{(1)}_{(+)}$, and $\mathbf{U}^{(2)}_{(+)}$.
2. Sample the global time of firing of the next reaction, $\tau^{\mathrm{g}}$:
   - 2.1. Generate a uniform random number $r_1$ in [0,1).
   - 2.2. If $\Delta == 0$ (i.e., $\mathbf{T}$ is empty) then $t \leftarrow t - \log r_1 / a$
   - 2.3. else
     - 2.3.1. $\lambda_1 \leftarrow t$; $\lambda_2 \leftarrow \mathrm{T}_1$; $a_t \leftarrow a(\lambda_2 - \lambda_1)$; $F \leftarrow 1 - \exp(-a_t)$
     - 2.3.2. While $F < r_1$
       - 2.3.2.1. Get current delay reaction and its type from $\mu_1^{(\mathrm{D})}$ and $\mathrm{D}_1$, respectively. Update $\boldsymbol{n}$, $\boldsymbol{\Pi}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\Sigma}$ accordingly using the proper subset of update structures $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(1)}_{(-)}$, $\mathbf{U}^{(2)}_{(-)}$, $\mathbf{U}^{(1)}_{(+)}$, and $\mathbf{U}^{(2)}_{(+)}$, $\mathbf{U}^{(3)}$ (see Section 6.2.1). Calculate $\delta a$ and set $a \leftarrow a + \delta a$.
       - 2.3.2.2. $\lambda_1 \leftarrow \mathrm{T}_1$. Remove $\mathrm{T}_1$, $\mu_1^{(\mathrm{D})}$, and $\mathrm{D}_1$ from the corresponding lists and decrement $\Delta \leftarrow \Delta - 1$.
       - 2.3.2.3. If $\Delta == 0$ then exit from the while loop 2.3.2.
       - 2.3.2.4. else $\lambda_2 \leftarrow \mathrm{T}_1$
       - 2.3.2.5. $a_t \leftarrow a_t + a(\lambda_2 - \lambda_1)$; $F \leftarrow 1 - \exp(-a_t)$
     - 2.3.3. if $\Delta == 0$ then $\tau^{\mathrm{g}} \leftarrow \lambda_1 + \frac{-\log(1-r_1) - a_t - a(\lambda_2 - \lambda_1)}{a}$; set $t \leftarrow \tau^{\mathrm{g}}$
     - 2.3.4. else $\tau^{\mathrm{g}} \leftarrow \lambda_1 + \frac{-\log(1-r_1) - a_t}{a}$; set $t \leftarrow \tau^{\mathrm{g}}$.
3. Sample the index of the next reaction, $\mu$: Using linear search, sample the group index $I$ and element index $J$ of the next reaction according to Eqs. 6.9 and 6.10, respectively. Look up the index of the next reaction as $\mu = \mathrm{L}_{I,J}$.
4. If $\mu$ is a delay reaction, increment $\Delta \leftarrow \Delta + 1$. Insert $t + d_\mu$ into $\mathbf{T}$, $\mu$ into $\boldsymbol{\mu}^{(\mathrm{D})}$, and the type of the delay reaction into $\mathbf{D}$. Use bisection search to ensure that the entries in $\mathbf{T}$ are in ascending order and maintain the correspondence between $\mathbf{T}$, $\boldsymbol{\mu}^{(\mathrm{D})}$, and $\mathbf{D}$.
4. Update $\boldsymbol{n}$ depending on reaction $\mu$'s type:
   - 4.1. If $\mu$ is $\mathrm{R}_{\mathrm{D}0}$, then update $\boldsymbol{n}$ using $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$
   - 4.2. else if $\mu$ is $\mathrm{R}_{\mathrm{D}1}$, then do not update $\boldsymbol{n}$
   - 4.2. else if $\mu$ is $\mathrm{R}_{\mathrm{D}2}$, then update $\boldsymbol{n}$ using $\mathbf{U}^{(1)}_{(-)}$ and $\mathbf{U}^{(2)}_{(-)}$
5. Update $\boldsymbol{\Pi}$ using the update structure $\mathbf{U}^{(3)}$ and calculate the change in total propensity $\delta a_0$.
6. Update $a \leftarrow a + \delta a$.
7. Go to step 2.

Table 6.2: Detailed algorithm for the delay partial-propensity direct method (dPDM), explicitly describing all sub-steps. Using the `multiset` container of the C++ STL, the list of pending reactions is conveniently maintained.

### 6.2.2 Computational cost

The computational cost of dPDM as detailed in Table 6.2 is determined by the following steps:

**Update step:** The computational cost of the update step is $O(N)$ and $O(1)$ for strongly and weakly coupled reaction networks, respectively, where $N$ is the number of species in the network.

**Sampling the global time of the next reaction:** The computational cost of sampling the global time of firing (initiation) of the next reaction, $\tau^{\mathrm{g}}$, is $O(pN)$ and $O(p)$ for strongly and weakly coupled reaction networks, respectively. This is because the number of times the partial propensities need to be updated due to a finishing pending reaction is $p$, where $p$ is search depth to locate $\tau^{\mathrm{g}}$. During each of these $p$ updates, the number of partial propensities that need to be recomputed is $O(N)$ and $O(1)$ for strongly and weakly coupled reaction networks, respectively. Removing the $p$ completed reactions from the list of pending reactions is $O(p)$.

**Sampling the index of the next reaction:** Sampling the group index is performed using linear search across at most $N+1$ groups. Subsequently, the element index is sampled using linear search across the $O(N)$ partial propensities within the selected group. The computational cost of sampling the index of the next reaction hence is $O(N)$. If the sampled reaction is a delay reaction, it is added to the list of pending reactions, along with its attributes. Inserting a new reaction such a way that the global completion times of pending reactions are maintained in ascending order is $O(\log_2 \Delta)$, where $\Delta$ is the number of pending reactions currently in the list.
In summary, the computational cost of dPDM is $O(pN + N + \log_2 \Delta)$ for strongly coupled reaction networks. This is equivalent to $O(pN + \log_2 \Delta)$ for $p > 0$. For weakly coupled reaction networks, the computational cost is $O(p + N + \log_2 \Delta)$. In general, the search depth $p$ is $O(\Delta)$. The worst case is realized when the time to the next reaction is past the last pending reaction. In this case, $p = \Delta$. The computational cost for subsequently sampling the next reaction is then $O(N)$, without the $O(\log_2 \Delta)$ term. This is because the queue of pending reactions is empty and cost of inserting the new pending reaction is $O(1)$. The overall cost of dPDM then is $O(\Delta N + N)$ and $O(\Delta + N)$ for strongly coupled and weakly coupled reaction networks, respectively.

### 6.2.3 Benchmarks

We benchmark the computational performance of dPDM on both a weakly coupled and a strongly coupled prototypical reaction network. We again choose the cyclic chain model [15, 47] and the colloidal aggregation model [35, 36, 37, 38, 39] as representative networks for which we compare the performance of dPDM with that of dDM [89]. In the benchmarks, we only consider consuming delay reactions since they require updates at both the time of reaction initiation as well as completion.
All tested SSA formulations are implemented in C++ using the random number generator of the GSL library and compiled using the Intel C++ compiler version 11.1 with the O3 optimization flag. All timings are measured on a Linux 2.6 workstation with a 2.8 GHz quad-core Intel Xeon E5462 processor, 8 GB of memory and 4 MB L2 cache. For all test cases, we simulate the reaction network until $10^7$ reactions have been initiated, and we report the average CPU time $\Theta$ per reaction initiation (i.e., the average time to execute steps 2 through 7 in Table 6.2 for dPDM and Table 6.1 for dDM).

#### 6.2.3.1 A strongly coupled reaction network: Colloidal aggregation model

The colloidal aggregation model is given in Eq. 2.36. For $N$ chemical species, the number of reactions is $M = \left\lfloor \frac{N^2}{2} \right\rfloor$. The degree of coupling of this reaction network is $3N-7$ and hence scales with system size.
At time $t = 0$, we set all $n_i = 1$ and all specific probability rates $c_\mu = 1$. We set all reactions with an even index to be consuming delay reactions ($\mathrm{R_{D2}}$), each with a delay of $d_\mu = 0.1$. The rest of the reactions are non-delay reactions ($\mathrm{R_{D0}}$). The benchmarks confirm that the search depth $p$ to sample the global time of firing (initiation) of the next reaction is $O(1)$, and that the logarithm of the number of pending delay reactions, $\log_2 \Delta$, is $O(\log_2 N)$. Hence, the computational cost of this simulation is $O(N)$ for dPDM and $O(N^2)$ for dDM. This is shown in Fig. 6.2A, where $\Theta(N)$ for dPDM and dDM are compared.
Figure 6.2B shows the results for larger networks on a linear scale. Here, we consider networks of up to $N = 2000$ species and $M = 2$ million reactions in order to reveal memory contention effects. Around $N = 1000$ species, the slope of the cost curve increases, while remaining $O(N)$. This is probably due to the partial-propensity structure not fitting into cache any more. The machine used for the benchmark has a 4 MB L2 cache. At $N = 1000$ the partial-propensity structure for this network contains 500 000 double-precision floating-point numbers of 8 bytes each, amounting to exactly 4 MB.
In summary, for a strongly coupled reaction network, the computational cost of dPDM is $O(pN + \log_2 \Delta)$ as predicted by the theoretical analysis.

#### 6.2.3.2 A weakly coupled reaction network: Cyclic chain model

The cyclic chain model is given by Eq. 2.44. For $N$ chemical species, this network has $M = N$ reactions. The degree of coupling of this reaction network is 2, independent of system size.
At time $t = 0$, we set all $n_i = 1$ and all specific probability rates $c_\mu = 1$. We set all reactions with an even index to be consuming delay reactions ($\mathrm{R_{D2}}$), each with a delay $d_\mu = 0.1$. The rest of the reactions are non-delay reactions ($\mathrm{R_{D0}}$). The benchmarks confirm that the search depth $p$ to sample $\tau^{\mathrm{g}}$ is $O(1)$ and that $\log_2 \Delta$ is $O(\log_2 N)$. Hence, the computational cost of this simulation is $O(N)$ for dPDM as well as for dDM. The corresponding $\Theta(N)$ for dPDM and dDM are shown in Fig. 6.2C.
In summary, for a weakly coupled reaction network, the computational cost of dPDM is $O(p + N + \log \Delta)$ as predicted by the theoretical analysis.

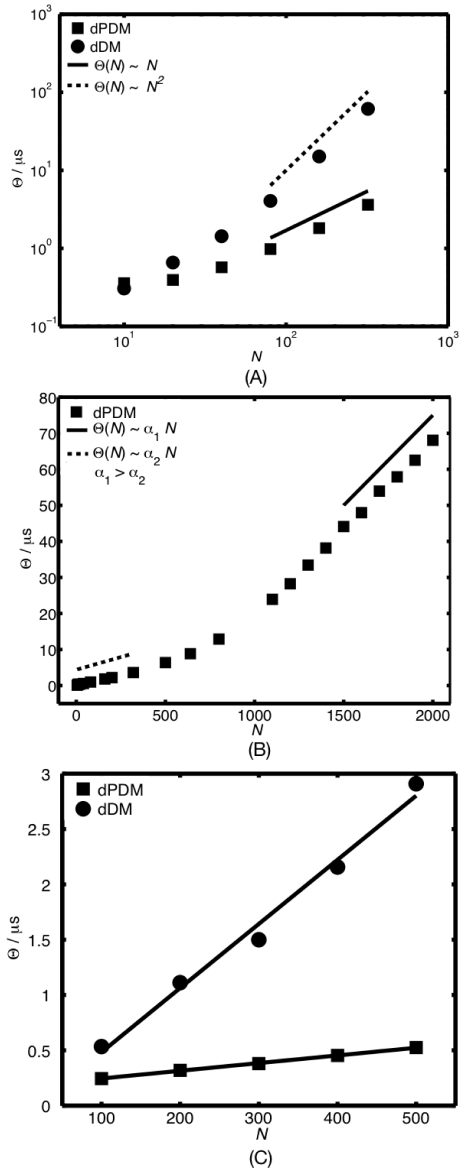



Figure 6.2: Computational cost of dPDM (squares) and dDM (circles). The average (over 100 independent runs) CPU time $\Theta$ per reaction initiation (i.e., the average time to execute steps 2 through 7 in Table 6.2 for dPDM, and Table 6.1 for dDM) is shown as a function of the number of species $N$ in the reaction network. (A) Logarithmic plot of $\Theta(N)$ for the strongly coupled colloidal aggregation model, considering systems of size up to $N = 320$. $\Theta$ is $O(N)$ for dPDM and $O(M) = O(N^2)$ for dDM. (B) Linear plot of $\Theta(N)$ for the strongly coupled colloidal aggregation model, considering systems of size up to $N = 2000$ (2 million reactions). While the scaling of the computational cost remains linear for all system sizes tested, the slope increases around $N = 1000$. This is the system size beyond which the partial-propensity structure does not fit into the computer's cache memory any more. (C) Linear plot of $\Theta(N)$ for the weakly coupled cyclic chain model. The solid lines are linear least square fits. $\Theta$ is $O(N)$ for both dPDM and dDM, but with a smaller slope for dPDM.

### 6.2.4 Conclusions

We have presented the delay partial-propensity direct method (dPDM), a partial-propensity formulation of the delay stochastic simulation algorithm (dSSA) [89] to simulate chemical reaction networks with delays. dPDM uses partial propensities and reaction groups in order to improve computational efficiency. For reaction networks with no delays, dPDM becomes identical to the partial-propensity direct method (PDM) (see Sec. 2.4.3).
The presented dPDM is an exact dSSA formulation with a computational cost of $O(pN+N+\log_2 \Delta)$ for strongly coupled reaction networks and $O(p+N+\log_2 \Delta)$ for weakly coupled networks. Here, $N$ is the number of chemical species, $p$ is the search depth to sample the time to the next reaction, and $\Delta$ is the number of pending delay reactions at a given time. We have presented a theoretical cost analysis of dPDM and confirmed its results in two benchmark cases prototypical of strongly and weakly coupled reaction networks. Since $p$ and $\Delta$ are properties of the chemical reaction network alone, and the only other variable that the computational cost depends on is linear in $N$, dPDM is especially efficient for strongly coupled reaction networks with delays. This is because in these networks the number of chemical species $N$ grows much slower with network size than the number of chemical reactions $M$.
However, dPDM inherits the limitations of PDM (see Sec. 2.4.4.7). Like PDM and PSSA-CR, it is limited to chemical reaction networks composed of elementary reactions involving at most two reactants. For small networks, dPDM is outperformed by other methods due to the overhead of the additional data structures. Other dSSA formulations, such as the delay direct method (dDM) [89], might be more efficient there.
The computational cost of dPDM can be further reduced to $O(p+\log_2 \Delta)$ for weakly coupled reaction networks by using composition-rejection sampling [30, 18] (see Sections 2.3.7 and 2.4.5) instead of linear search [2] to sample the index of the next reaction. This is analogous to PSSA-CR. For multi-scale (stiff) reaction networks, prototypical of biochemical networks where the propensities span several orders of magnitude, dynamic sorting [16] (see Sec. 2.4.4) can further reduce the computational cost, even though its scaling with $N$ remains the same.

# 7

# Spatiotemporal Simulation of Stochastic Reaction-Diffusion Systems

*In this chapter:*

- Off-lattice and on-lattice approaches to stochastic reaction-diffusion simulation
- The Next-Subvolume Method for exact reaction-diffusion simulations
- Discretization correction of reaction propensities
- Partial-propensity methods for exact stochastic reaction-diffusion simulations
- Computational cost analysis of simulation methods with respect to network size and spatial resolution

*Learning goals:*

- Be able to explain how on- and off-lattice reaction-diffusion simulations work and what their advantages and problems are
- Be able to implement the next-subvolume method
- Be able to implement propensity corrections in cubic domains and know about their necessity
- Be able to use partial propensities and composition-rejection sampling to speed up stochastic reaction-diffusion simulations
- Be able to derive or explain the algorithmic complexity of PSRD

Chemical reaction networks exhibiting spatial heterogeneity are often modeled as reaction-diffusion systems [96, 97, 98, 99, 100, 101, 102, 103, 104, 105]. Reaction-diffusion models explicitly capture spatial variations of the concentration fields, accounting for diffusive transport of reactants and products to and from reaction sites. Spatial heterogeneity is sustained when diffusion of chemicals is slower than reactions between them. In the limit of large numbers of molecules, reaction-diffusion processes can be modeled continuously as systems of coupled partial differential equations (frequently called *reaction-diffusion equations* or *Fisher-KPP equations* [96, 97]) governing the spatiotemporal evolution of the smooth concentration fields of all chemical species. Continuum reaction-diffusion models can exhibit nontrivial spatiotemporal dynamics, such as traveling concentration fronts [106] and inhomogeneous stationary concentration distributions ("Turing patterns") [107, 108, 98, 109]. These phenomena have been successful in explaining a number of experimental observations, including localization of cell division sites in *E. coli* [110] and "black eyes" patterns in the chlorite-iodide-malonic acid reaction [111, 112, 113]. For low molecular copy numbers, however, continuum models fail to provide an accurate description of the spatiotemporal dynamics of reaction-diffusion systems. In particular, intrinsic noise from the apparent molecular discreteness, leading to stochasticity of chemical reactions, alters front propagation dynamics [114] and Turing patterns [100, 101] in a nontrivial way. This is because fluctuations in the molecule populations may no longer be negligible, and correlated fluctuations may lead to deviations from deterministic behavior [115, 100, 101, 103, 116]. These effects can be accounted for by stochastic reaction diffusion (SRD) simulations.

There are mainly two types of SRD simulations: on-lattice (or compartment-based) simulations and off-lattice (or particle-based) simulations. On-lattice simulations include the Next Subvolume Method (NSM) [100], whereas Greens-Function Reaction Dynamics (GFRD) [117] and Brownian Dynamics (BD) [118] are examples of off-lattice schemes. On-lattice SRD simulations [100, 119, 120, 121, 122, 123, 124] are based on dividing (discretizing) the computational domain into subvolumes, in each of which the chemical reaction system is assumed to be well mixed (spatially homogeneous). It is further postulated that only molecules within the same subvolume can react with each other, effectively treating molecules of the same chemical in different subvolumes as different species. Diffusion is modeled as unimolecular "diffusion reactions" representing jumps of molecules between neighboring subvolumes. The on-lattice approach hence describes the reaction-diffusion system as a large chemical reaction network with the number of species proportional to the product of the actual number of chemical species and the number of subvolumes used to discretize space. The kinetics of this enlarged reaction network can be mathematically described by the on-lattice reaction-diffusion master equation (RDME), analogous to the chemical master equation (CME) [3]. Off-lattice SRD simulations [118, 117, 125, 126] are based on computational particles mimicking the Brownian motion of molecules, whereby the molecules involved in a bimolecular reaction react with a certain probability when the distance between them is smaller than a pre-defined reaction radius.

Here, we focus on-lattice SRD simulations in order to avoid computationally expensive collision detection and time-step adaptation mechanisms [127]. Since on-lattice SRD is described by a system of chemical reactions modeled by the RDME, it can be exactly simulated using Gillespie's stochastic simulation algorithm (SSA) [2, 3]. SSA samples trajectories from the exact solution of the master equation by sampling the index of the next reaction, the time to the next reaction, and updating the reaction probability rates (called "propensities"). Different SSA formulations are available that use different sampling and update algorithms, including the direct method (DM) [2, 14], the first reaction method (FRM) [2, 14], the next reaction method (NRM) [1], the optimized direct

method (ODM) [15], the sorting direct method (SDM) [16], the SSA with composition-rejection sampling (SSA-CR) [18] (see Sec. 2.3), and partial-propensity methods (see Chapter 2.4) such as the partial-propensity direct method (PDM) (Sec. 2.4.3), the sorting PDM (SPDM) (Sec. 2.4.4), and the partial-propensity SSA with composition-rejection sampling (PSSA-CR) (Sec. 2.4.5). Directly using any of these SSA formulations for the RDME, without adapting it to the specifics of on-lattice SRD simulations, would be correct, but computationally and/or memory inefficient since the number of species and the number of reactions increase linearly with the number of subvolumes. A more efficient way of performing on-lattice SRD simulation is to first sample the subvolume in which the next reaction will happen and then sample the index of the reaction within that subvolume. This is, for example, done in NSM [100] as implemented in the MesoRD software package [119]. NSM uses NRM for sampling the subvolume and DM for sampling the reaction within that subvolume [100, 119]. For a chemical reaction network with $N$ species and $M$ reactions, the computational cost of NSM to perform an on-lattice SRD simulation in a three-dimensional (3D) computational domain divided into $N_\mathrm{v}$ subvolumes is $O(\log_2 N_\mathrm{v} + M f_\mathrm{r} + 6N(1-f_\mathrm{r}))$ [100, 119], where $f_\mathrm{r}$ is the fraction of firings accounted for by "real" reactions and $(1-f_\mathrm{r})$ the fraction of firings of "diffusion reactions". This is composed of the $O(\log_2 N_\mathrm{v})$ cost for maintaining the subvolume priority queue and the $O(M f_\mathrm{r} + 6N(1-f_\mathrm{r}))$ cost for sampling the next reaction. $M + 6N$ is the number of reactions in each subvolume, composed of the $M$ "real" chemical reactions and the $6N$ "diffusion reactions" to the 6 face-connected neighboring subvolumes in a uniform Cartesian 3D mesh (in 2D this would be $4N$).

If the molecular population increases, the time step of exact SSAs decreases, increasing the runtime of the simulations. This can be alleviated by approximate SSAs that use a fixed time step to sample trajectories from an approximate solution of the master equation. In this spirit, on-lattice SRD simulations involving larger population sizes can be accelerated using approximate and hybrid SSAs [120, 121, 122, 123, 124]. Here, we focus on exact on-lattice SRD formulations since they are parameter free and do not require prescription of a time step size or a target error level.

The idea of partial propensities can also be extended to spatiotemporal SRD simulations, leading to the partial-propensity SRD method, or PSRD. For weakly coupled reaction networks, where the number of reactions influenced by any other reaction is asymptotically independent of system size, the computational cost of partial-propensity methods is bounded by a constant (see Chapter 2.4). This is particularly advantageous for on-lattice SRD simulations, where the number of neighboring subvolumes influenced by any subvolume is constant (6 in 3D, 4 in 2D), independent of the total number of subvolumes used to discretize space. PSRD hence uses composition-rejection sampling to find the next subvolume with an $O(G_\mathrm{a})$ cost, and then uses SPDM inside that subvolume to sample the next reaction with a cost of $O(N)$. $G_\mathrm{a}$ is the logarithm of the ratio of the maximum to the minimum non-zero subvolume propensities, which is at most $O(\log_2 N_\mathrm{v})$. PSRD thus has an overall computational cost of $O(G_\mathrm{a} + N)$, which is asymptotically bounded from above by $O(\log_2 N_\mathrm{v} + N)$ and independent of the ratio between "real" and "diffusion" reactions. This is achieved by restricting the method to elementary chemical reactions, under the premise that any non-elementary reactions can be broken down into elementary reactions [31, 32, 3] at the expense of an increase in network size. We demonstrate the scaling of the computational cost of PSRD on two types of reaction networks: one in which the number of reactions $M$ increases super-linearly with the number of species $N$ (a strongly coupled network), and a second in which $M$ is proportional to $N$ (a weakly coupled network). Finally, we demonstrate the application of PSRD to pattern-forming stochastic Gray-Scott systems [128, 129, 130, 98, 131], highlighting the effect of intrinsic noise on the resulting Turing patterns.

## 7.1 On-lattice stochastic reaction-diffusion

We introduce on-lattice SRD simulations using an example reaction network. In the benchmarks presented below we assume that the boundary of the computational domain is reflective (no-flux boundary condition), except for the showcases in Sec. 7.2.5, where we use periodic boundary conditions. Other boundary conditions can be treated as described by Erban and Chapman (2007) [132] (see Sec. 2 in their article). The scaling of the computational cost of on-lattice SRD simulations, however, is independent of the type of the boundary condition.

### 7.1.1 General concept

Consider the example of the following trimerization reaction in a 3D cuboidal reactor of dimension $L_x \times L_y \times L_z$ and volume $\Omega = L_x L_y L_z$:

$$\begin{array}{lccc} \text{Reaction 1} : & \emptyset & \xrightarrow{k_1} & \mathrm{S}_1 \\ \text{Reaction 2} : & \mathrm{S}_1 + \mathrm{S}_1 & \xrightarrow{k_2} & \mathrm{S}_2 \\ \text{Reaction 3} : & \mathrm{S}_1 + \mathrm{S}_2 & \xrightarrow{k_3} & \mathrm{S}_3 \\ \text{Reaction 4} : & \mathrm{S}_3 & \xrightarrow{k_4} & \emptyset. \end{array} \tag{7.1}$$

The $k$'s are the macroscopic reaction-rate constants. This reaction network has $N = 3$ species and $M = 4$ reactions. We choose this reaction network as an example since it contains all types of elementary reactions: reaction 1 is a source reaction, reaction 2 a bimolecular reaction between the same species (homo-bimolecular reaction), reaction 3 a bimolecular reaction between two different species (hetero-bimolecular reaction), and reaction 4 is a unimolecular reaction. Any non-elementary reaction involving $r > 2$ reactants can be broken down to a set of $2r - 3$ elementary reactions by introducing additional $r - 2$ auxiliary species. The reaction-propensity $a_\mu$ of reaction $\mu$ is defined as the probability rate of firing of that reaction. Each $a_\mu$ is computed as the product of the reaction degeneracy and the specific probability rate $c_\mu$ of that reaction. The reaction degeneracy is the number of distinct combinations (collision pairs) of reactant molecules that can be formed, and the specific probability rate is the probability rate of the reaction when only one molecule of each reactant is present. According to these definitions, the reaction propensities for the reaction network in Eq. 7.1 are:

$$a_\mu = \begin{cases} c_\mu, & c_\mu = k_1\Omega, & \text{if } \mu = 1 \\ \frac{1}{2} n_1(n_1 - 1)c_\mu, & c_\mu = 2k_2\Omega^{-1}, & \text{if } \mu = 2 \\ n_1 n_2 c_\mu, & c_\mu = k_3\Omega^{-1}, & \text{if } \mu = 3 \\ n_3 c_\mu, & c_\mu = k_4, & \text{if } \mu = 4\,, \end{cases} \tag{7.2}$$

where $n_i$ denotes the population of species $\mathrm{S}_i$, i.e., the number of molecules of $\mathrm{S}_i$ present in the system.

If the characteristic time of diffusion of the species is comparable to or larger than the characteristic time of reaction, the system will exhibit spatial inhomogeneities and diffusion of the species in the reaction network needs to be explicitly accounted for. In on-lattice SRD methods, this is done by dividing the computational domain into subvolumes within which the system is assumed to be well mixed. The chemical species in each subvolume can (i) react with each other in bimolecular reactions, (ii) undergo unimolecular reactions, or (iii) be produced through source reactions. In both cases, the products are formed in the same subvolume and species from different subvolumes

can not react with each other. Diffusion of molecules is modeled as a jump process from a subvolume to any of the face-connected neighboring subvolumes.

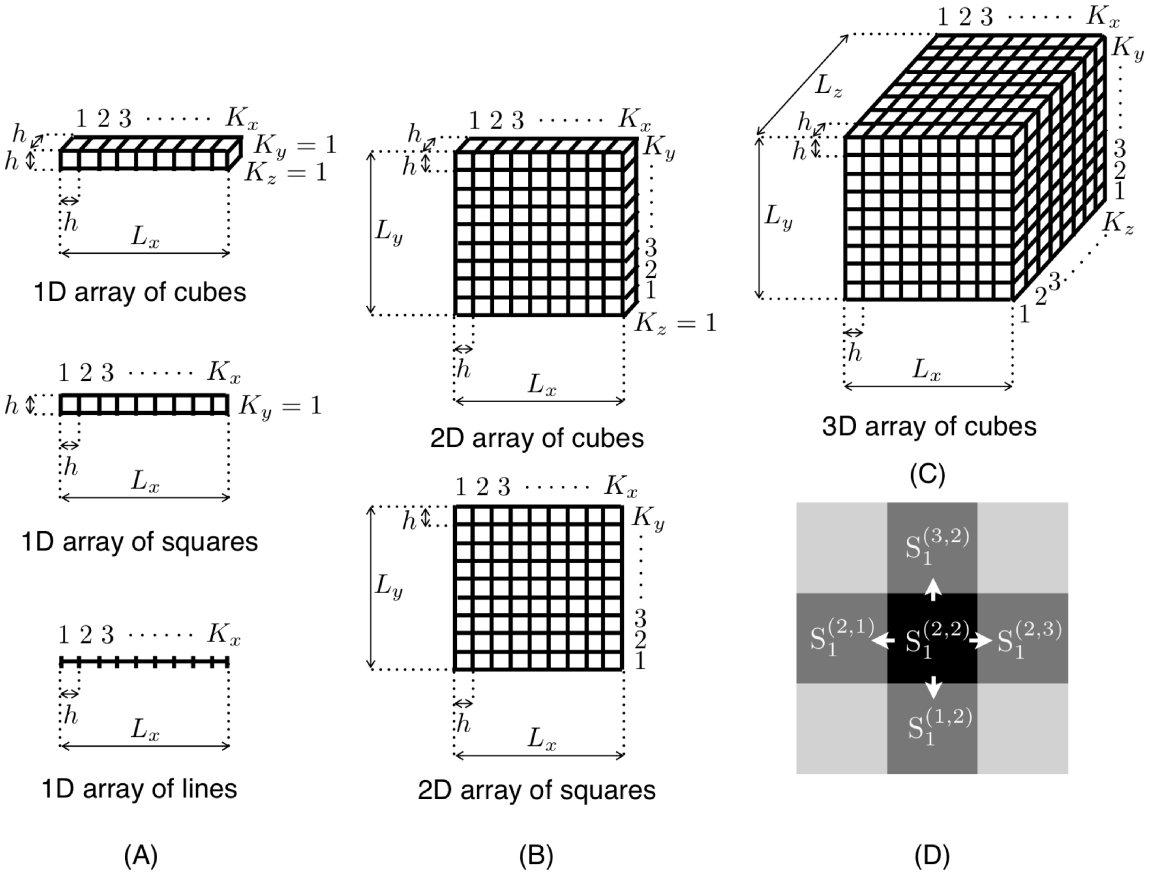



Figure 7.1: Division of the computation domain into subvolumes. (A–C) Different possibilities of a subdiving a box-shaped computational domain in one, two, and three dimensions, respectively. $L_x$, $L_y$, and $L_z$ are the lengths of the computational domain in each direction. $K_x$, $K_y$, and $K_z$ are the numbers of subvolumes of edge length $h$ after subdivision in each direction. (D) Diffusion is modeled as jump "reactions" to face-connected subvolumes. The same chemical in different subvolumes is treated as a different species in stochastic on-lattice reaction-diffusion simulations. Unimolecular "diffusion reactions" convert species as shown.

Assume that we divide the 3D computational domain into $N_\mathrm{v} = K_x K_y K_z$ equi-sized cubic subvolumes of edge length $h = L_x/K_x = L_y/K_y = L_z/K_z$ and volume $\Omega_\mathrm{c} = h^3$ (see Fig. 7.1C; the one- and two-dimensional cases are illustrated in Figs. 7.1A and 7.1B, respectively). The subvolumes are indexed by their Cartesian mesh coordinates over the set [133]

$$I_\mathrm{all} = \{(l,m,n) \mid l,m,n \text{ are integers such that } 1 \le l \le K_x\,;\, 1 \le m \le K_y\,;\, 1 \le n \le K_z\,\} \qquad (7.3)$$

and the set of face-connected neighbors of a subvolume with index $(l, m, n)$ is

$$C_{l,m,n} = \{(l, m, n) + \boldsymbol{\delta} \mid (l, m, n) + \boldsymbol{\delta} \in I_{\text{all}}\} \tag{7.4}$$

such that $\boldsymbol{\delta} \in E = \{(1,0,0), (-1,0,0), (0,1,0), (0,-1,0), (0,0,1), (0,0,-1)\}$. Hence, the on-lattice reaction-diffusion system of the reaction network in Eq. 7.1 can be written as:

$$\left.\begin{array}{ccccc}
\emptyset & \xrightarrow{k_1^{(l,m,n)}} & \mathrm{S}_1^{(l,m,n)}, & a_1^{(l,m,n)} \\
\mathrm{S}_1^{(l,m,n)} + \mathrm{S}_1^{(l,m,n)} & \xrightarrow{k_2^{(l,m,n)}} & \mathrm{S}_2^{(l,m,n)}, & a_2^{(l,m,n)} \\
\mathrm{S}_1^{(l,m,n)} + \mathrm{S}_2^{(l,m,n)} & \xrightarrow{k_3^{(l,m,n)}} & \mathrm{S}_3^{(l,m,n)}, & a_3^{(l,m,n)} \\
\mathrm{S}_3^{(l,m,n)} & \xrightarrow{k_4^{(l,m,n)}} & \emptyset, & a_4^{(l,m,n)} \\
\mathrm{S}_1^{(l,m,n)} & \xrightarrow{D_1/h^2} & \mathrm{S}_1^{\mathbf{c}}, & a_5^{(l,m,n)} \\
\mathrm{S}_2^{(l,m,n)} & \xrightarrow{D_2/h^2} & \mathrm{S}_2^{\mathbf{c}}, & a_6^{(l,m,n)} \\
\mathrm{S}_3^{(l,m,n)} & \xrightarrow{D_3/h^2} & \mathrm{S}_3^{\mathbf{c}}, & a_7^{(l,m,n)}
\end{array}\right\} \quad \forall (l, m, n) \in I_{\text{all}},\ \mathbf{c} \in C_{l,m,n}, \tag{7.5}$$

where $\mathrm{S}_i^{(l,m,n)}$ denotes species $\mathrm{S}_i$ in subvolume $(l, m, n)$, the $k$'s the macroscopic reaction rates, and the $a$'s the corresponding propensities. In general, the $k$'s can be different in different subvolumes, which is explicitly shown in Eq. 7.5 by indexing them with the subvolume index. Diffusion of species $\mathrm{S}_i$ with diffusion constant $D_i$ is modeled as jumps to face-connected neighboring subvolumes as illustrated in Fig. 7.1D. Equation 7.5 models the on-lattice reaction-diffusion system as a system of chemical reactions composed of $3K_xK_yK_z = 3N_\mathrm{v}$ species and $22K_xK_yK_z - 6(K_xK_y + K_yK_z + K_xK_z) = 22N_v - 6N_\mathrm{v}(\frac{1}{K_x} + \frac{1}{K_y} + \frac{1}{K_z})$ reactions composed of $4K_xK_yK_z$ "real" reactions and $3(6K_xK_yK_z - 6(K_xK_y + K_yK_z + K_xK_z))$ "diffusion reactions", accounting for the missing neighboring subvolumes at the domain boundary. In general, the 3D SRD dynamics of $N$ species and $M$ reactions in a computational domain with reflective boundaries and $K_x \times K_y \times K_z$ subvolumes can be modeled by a chemical reaction network consisting of $NK_xK_yK_z = NN_\mathrm{v}$ species and $(M+6N)K_xK_yK_z - 2(K_xK_y + K_yK_z + K_xK_z)N = (M+6N)N_\mathrm{v} - 2\left(\frac{1}{K_x} + \frac{1}{K_y} + \frac{1}{K_z}\right)N_\mathrm{v}N$ reactions. For other boundary conditions, the number of reactions is $(M+6N)N_\mathrm{v}$, accounting for the diffusive fluxes across the boundary.

For inhomogeneous diffusion, $D_i$ additionally depends on the subvolume index $(l, m, n)$. For anisotropic diffusion, $D_i$ depends on the direction of the jump reaction. These extensions are straightforward to include in any on-lattice SRD framework.

The propensities of the reactions in Eq. 7.5 are:

$$a_\mu^{(l,m,n)} = \begin{cases}
c_\mu^{(l,m,n)}, & c_\mu^{(l,m,n)} = k_1^{(l,m,n)}\Omega_\mathrm{c}, & \text{if } \mu = 1 \\
\frac{1}{2}n_1^{(l,m,n)}(n_1^{(l,m,n)} - 1)c_\mu^{(l,m,n)}, & c_\mu^{(l,m,n)} = 2k_2^{(l,m,n)}\Omega_\mathrm{c}^{-1}, & \text{if } \mu = 2 \\
n_1^{(l,m,n)}n_2^{(l,m,n)}c_\mu^{(l,m,n)}, & c_\mu^{(l,m,n)}k_3^{(l,m,n)}\Omega_\mathrm{c}^{-1}, & \text{if } \mu = 3 \\
n_3^{(l,m,n)}c_\mu^{(l,m,n)}, & c_\mu^{(l,m,n)} = k_4^{(l,m,n)}, & \text{if } \mu = 4 \\
n_1^{(l,m,n)}c_\mu^{(l,m,n)}, & c_\mu^{(l,m,n)} = D_1h^{-2}, & \text{if } \mu = 5 \\
n_2^{(l,m,n)}c_\mu^{(l,m,n)}, & c_\mu^{(l,m,n)} = D_2h^{-2}, & \text{if } \mu = 6 \\
n_3^{(l,m,n)}c_\mu^{(l,m,n)}, & c_\mu^{(l,m,n)} = D_3h^{-2}, & \text{if } \mu = 7\,,
\end{cases} \tag{7.6}$$

where $n_i^{(l,m,n)}$ is the population of species $\mathrm{S}_i^{(l,m,n)}$ (i.e., species $\mathrm{S}_i$ in subvolume $(l,m,n)$) and $c_\mu^{(l,m,n)}$ is the specific probability rate of reaction $\mu$ in subvolume $(l,m,n)$. These formulations for the propensities directly follow from the same argument as the propensities in Eq. 7.2 for the reaction system given in Eq. 7.1. The rates of the "diffusion reactions" always scale as $h^{-2}$, irrespective of the dimension of the subvolumes.

### 7.1.2 Discretization-corrected propensities

The propensity formulations in Eq. 7.6 may lead to artifacts in the kinetics introduced by the spatial discretization [133, 127]. This is due to the subdivision of the reaction space into several small subvolumes. This subdivision is fundamentally different from the one used in spatial discretization of continuum models (e.g., finite difference or finite volume methods). While in discretizing continuum models more resolution (smaller subvolumes) is always better, this is not necessarily the case in SRD simulations. This is because in SRD simulations, the subvolumes introduce spurious physical boundaries; molecules in one subvolume cannot react with molecules in a neighboring subvolume, even if for molecules close to a subvolume boundary the closest collision partner could be in a neighboring subvolume. The subvolumes thus define closed, well-mixed reaction spaces of volume $\Omega_c \ll \Omega$. In order for the reaction system to be well mixed within each subvolume, the subvolume edge lengths have to be much smaller than the Kuramoto length [134], hence

$$h \ll h_{\max} = \sqrt{2dDt_{\mathrm{r}}}\,, \tag{7.7}$$

where $d$ is the dimension of the subvolume. The characteristic time $t_{\mathrm{r}}$ of the fastest reactions in the system can be estimated from the time autocorrelation function of species populations simulated using exact SSA. At length scales above $h_{\max}$ the subvolumes are no longer spatially homogeneous (well mixed).

In addition to this upper bound on $h$, there may also be a lower bound. It is, for example, known that chemical kinetics in small volumes is quantitatively and qualitatively altered [135, 136, 73]. It has further been shown that the RDME gives different results depending on the level of spatial discretization of the computational domain [133]. If the discretization becomes too fine, the RDME even yields unphysical results [133, 127]. These artifacts are introduced by the artificial subdivision of space and lead to propensities in the RDME becoming inconsistent with Smoluchowski's microscopic reaction-diffusion framework [35, 127]. The propensities in on-lattice SRD simulations hence need to be corrected for the spatial discretization.

There are two strategies toward deriving discretization-corrected propensities. The first is based on the premise that for a well-stirred system of reactions the kinetics of the reaction-diffusion process should not depend on the resolution of the spatial discretization [133]. In this strategy, only the propensities of bimolecular reactions need to be corrected. Assuming a 3D cubic computational domain of size $L \times L \times L$ that is divided into $K^3$ subvolumes of edge length $h = L/K$, Erban and Chapman (2009) [133] have derived discretization-corrected propensities for bimolecular reactions. For a hetero-bimolecular reaction $\mu$ occurring in subvolume $(l,m,n)$

$$\mathrm{S}_i^{(l,m,n)} + \mathrm{S}_j^{(l,m,n)} \quad \xrightarrow{k_\mu^{(l,m,n)}} \quad \text{Products} \tag{7.8}$$

the discretization-corrected propensity is given by:

$$a_\mu^{(l,m,n)} = n_i^{(l,m,n)} n_j^{(l,m,n)} c_\mu^{(l,m,n)}, \quad c_\mu^{(l,m,n)} = \frac{(D_i+D_j)k_\mu^{(l,m,n)}}{(D_i+D_j)h^3 - \beta k_\mu^{(l,m,n)} h^2}. \tag{7.9}$$

For a homo-bimolecular reaction $\mu$ occurring in subvolume $(l, m, n)$

$$\mathrm{S}_i^{(l,m,n)} + \mathrm{S}_i^{(l,m,n)} \xrightarrow{k_\mu^{(l,m,n)}} \text{Products} \tag{7.10}$$

the discretization-corrected propensity is given by:

$$a_\mu^{(l,m,n)} = \frac{n_\mu^{(l,m,n)}(n_\mu^{(l,m,n)}-1)}{2} c_\mu^{(l,m,n)}, \quad c_\mu^{(l,m,n)} = \frac{D_i k_\mu^{(l,m,n)}}{D_i h^3 - \beta k_\mu^{(l,m,n)} h^2}. \tag{7.11}$$

In Eqs. 7.9 and 7.11, the factor $\beta$ is given by:

$$\beta = \frac{1}{2K^3} \sum_{l,m,n=0;\ (l,m,n)\neq(0,0,0)}^{K-1} \frac{1}{3 - \cos(l\pi/K) - \cos(m\pi/K) - \cos(n\pi/K)}. \tag{7.12}$$

This discretization-correction framework imposes a lower bound on the admissible subvolume size, given by the constraint that the corrected reaction propensities have to be non-negative [133]. The RDME with corrected propensities is only physically valid for

$$h > h_{\min} = \max_\mu \left[ \frac{\beta k_\mu^{(l,m,n)}}{D_i + D_j}(1 - \delta_{ij}) + \frac{\beta k_\mu^{(l,m,n)}}{D_i}\delta_{ij} \right], \tag{7.13}$$

where $\delta_{ij}$ is the Kronecker delta. The maximum is taken over all bimolecular reactions $\mu$ where $D_i$ and $D_j$ are the diffusion coefficients of the two respective reactants.
The second strategy derives discretization-corrected propensities such that the RDME becomes consistent with Smoluchowski's microscopic reaction-diffusion framework [35, 127]. In this strategy, the discretization-corrected propensities depend on the population of reactant molecules in the neighboring subvolumes, rendering the correction non-local and reaction-dependent. This approach is valid also for non-cubic computational domains and non-3D simulations, and it does not impose any lower bound on the subvolume size $h$. It has been shown to provide a seamless transition between Smoluchowski's microscopic framework and that of on-lattice SRD as based on the RDME [35, 127].
In summary, the propensities of all reactions in a system obtained by on-lattice discretization of a reaction-diffusion process need to be corrected. Erban and Chapman (2009) [133] have derived the discretization correction only for cubic computational domains where the number of subvolumes in each spatial dimension is the same. We thus use the above discretization-corrected propensities only in these cases. Extending the present on-lattice SRD method to the framework proposed by Fange et al. (2010) [127] should be possible. The scaling of the computational cost of on-lattice SRD methods, however, is independent of the formulation used for the propensities.
We note that similar corrections are also necessary in off-lattice SRD simulations, where the artificial spatial discretization is, e.g., introduced by the reaction radius [133].

### 7.1.3 The Next Subvolume Method (NSM) for on-lattice stochastic reaction-diffusion simulations

NSM simulates the on-lattice SRD system by sampling from the conditional joint probability distribution function (PDF) for the time $\tau$ to the next reaction, the index $\mu$ of the next reaction, and the subvolume $(l, m, n)$ containing the next reaction, given the current population $\boldsymbol{n}(t)$ at time $t$. This joint PDF results from the on-lattice RDME and is given by:

$$p(\tau, \mu, l, m, n \,|\, \boldsymbol{n}(t)) = p(\tau)\ p(l, m, n)\ p(\mu \,|\, l, m, n)\,. \tag{7.14}$$

Here, $p(\tau)$ is the continuous PDF for the time to the next reaction, $\tau$, given by:

$$p(\tau) = a\mathrm{e}^{a\tau}, \tag{7.15}$$

where $a$ is the total propensity of the system. The discrete PDF $p(l, m, n)$ for the subvolume $(l, m, n)$ of the next reaction is given by:

$$p(l, m, n) = \frac{a^{(l,m,n)}}{a}, \tag{7.16}$$

where $a^{(l,m,n)}$ is the propensity of subvolume $(l, m, n)$. The discrete PDF $p(\mu \,|\, l, m, n)$ for the next reaction $\mu$ within subvolume $(l, m, n)$ is given by

$$p(\mu \,|\, l, m, n) = \frac{a_\mu^{(l,m,n)}}{a^{(l,m,n)}}, \tag{7.17}$$

$a_\mu^{(l,m,n)}$ the propensity of reaction $\mu$ in subvolume $(l, m, n)$. Formally, $\boldsymbol{n}(t) = \left[n_1^{(1,1,1)}, \ldots, n_N^{(1,1,1)}, \ldots, n_1^{(K_x,K_y,K_z)}, \ldots, n_N^{(K_x,K_y,K_z)}\right](t)$, where $n_i^{(l,m,n)}(t)$ is the population of species $\mathrm{S}_i$ in subvolume $(l, m, n)$ at time $t$, $a^{(l,m,n)} = \sum_\mu a_\mu^{(l,m,n)}$ the total propensity of all reactions in subvolume $(l, m, n)$, and $a = \sum_{l=1}^{K_x} \sum_{m=1}^{K_y} \sum_{n=1}^{K_z} a^{(l,m,n)}$ the total propensity of all reactions across all subvolumes.

NSM [100] is a popular and efficient algorithm for sampling trajectories of $\boldsymbol{n}(t)$ from the above PDF, which is the exact solution of the RDME. In NSM, the subvolume $(l, m, n)$ in which the next reaction will occur is sampled first according to Eq. 7.16 and subsequently one of the reactions $\mu$ in that subvolume is sampled according to Eq. 7.17. The latter is done by first deciding whether the next reaction is a "real" or a "diffusion" reaction and then using linear search only over the corresponding reaction group [119]. The algorithm used in NSM to sample the next subvolume is inspired by the indexed priority queues used in the next reaction method [1]. Sampling a reaction within a subvolume is done using linear search as in Gillespie's original direct method [2, 14]. The time to the next reaction is calculated from Eq. 7.15. After the chosen reaction fired, the population and the propensities of some of the reactions need to be updated. In NSM, the population is updated using a sparse representation of the stoichiometry matrix, and the propensities are updated using a dependency graph [1].

The computational cost of NSM is: (i) $O(1)$ for sampling the subvolume; (ii) $O(Mf_\mathrm{r} + 6N(1 - f_\mathrm{r}))$ for sampling the next reaction within that subvolume, where $M$ is the number of "real" reactions, $6N$ the number of "diffusion reactions" ($4N$ in 2D), and $f_\mathrm{r}$ the fraction of reaction firings accounted for by "real" reactions; (iii) $O(1)$ for updating the population; (iv) at most $O(M)$ for updating the propensities within a subvolume; (v) $O(\log_2 N_\mathrm{v})$ for updating the subvolume priority queue, where $N_\mathrm{v}$ is the number of subvolumes. The overall computational cost of NSM thus is $O(\log_2 N_\mathrm{v} + Mf_\mathrm{r} + 6N(1 - f_\mathrm{r}))$. The fraction $f_\mathrm{r}$ of "real" reaction firings decreases with increasing $N_\mathrm{v}$. For small $N_\mathrm{v}$, almost all reactions are "real" and the computational cost of NSM is $O(\log_2 N_\mathrm{v} + M)$. In particular, for $N_\mathrm{v} = 1$ the fraction $f_\mathrm{r} = 1$ and the computational cost of NSM is $O(M)$, as for Gillespie's DM [2, 14]. For large $N_\mathrm{v}$, the computational cost of NSM is $O(\log_2 N_\mathrm{v} + 6N)$ since $f_\mathrm{r} \ll 1$ and almost all reaction events pertain to "diffusion reactions".

## 7.2 The partial-propensity stochastic reaction-diffusion method (PSRD)

Combining ideas from NSM and partial-propensity SSAs (Chapter 2.4), we introduce a novel on-lattice SRD simulation method, the partial-propensity stochastic reaction-diffusion method (PSRD). PSRD is based on the idea of binning the subvolumes and determining the next subvolume using composition-rejection sampling [30, 18] (see Sections 2.3.7 and 2.4.5). Then, we use the concept of partial propensities (see Sec. 2.4.2) to sample the index of the next reaction within the selected subvolume.

### 7.2.1 General concept of PSRD

We summarize the general concepts of binned composition-rejection sampling and partial propensities in the context of SRD simulations.

#### 7.2.1.1 Composition-rejection sampling to select the subvolume

Composition-rejection sampling [30, 137, 138, 139] is an efficient algorithm to sample realizations of a random variable according to a given discrete probability distribution. In on-lattice SRD simulations, the discrete PDF for the subvolume index $(l,m,n)$ is $p(l,m,n)$ (see Eq. 7.16). The sampling process starts by binning the $a^{(l,m,n)}$ according to their values and then proceeds in two steps: The composition step is used to identify the bin by linear search, and the rejection step is used to identify the $a^{(l,m,n)}$, and hence the index of the subvolume $(l,m,n)$, inside that bin.

#### 7.2.1.2 Partial propensities to sample the next reaction within a subvolume

**Partial propensities:** The partial propensity of a reaction is defined as the propensity per molecule of one of its reactants (see Sec. 2.4.1). For example, the partial propensity $\pi_\mu^{(l,m,n);(i)}$ of reaction $\mu$ within a subvolume $(l,m,n)$ with respect to (perhaps the only) reactant $\mathrm{S}_i^{(l,m,n)}$ is $a_\mu^{(l,m,n)}/n_i^{(l,m,n)}$, where $a_\mu^{(l,m,n)}$ is the propensity of reaction $\mu$ in subvolume $(l,m,n)$ and $n_i^{(l,m,n)}$ the population of $\mathrm{S}_i^{(l,m,n)}$ (i.e., the number of molecules of species $\mathrm{S}_i$ in subvolume $(l,m,n)$). The partial propensities of the three elementary reaction types within each subvolume $(l,m,n)$ are:

- Bimolecular reactions $\mathrm{S}_i^{(l,m,n)} + \mathrm{S}_j^{(l,m,n)} \xrightarrow{c_\mu^{(l,m,n)}}$ Products: $a_\mu^{(l,m,n)} = n_i^{(l,m,n)}\, n_j^{(l,m,n)}\, c_\mu^{(l,m,n)}$ and $\pi_\mu^{(l,m,n);(i)} = n_j^{(l,m,n)}\, c_\mu^{(l,m,n)}$, $\pi_\mu^{(l,m,n);(j)} = n_i^{(l,m,n)}\, c_\mu^{(l,m,n)}$. If both reactants are of the same species, i.e. $\mathrm{S}_i^{(l,m,n)} = \mathrm{S}_j^{(l,m,n)}$, only one partial propensity exists, $\pi_\mu^{(l,m,n);(i)} = \frac{1}{2}(n_i^{(l,m,n)} - 1)c_\mu^{(l,m,n)}$, because the reaction degeneracy is $\frac{1}{2}n_i^{(l,m,n)}(n_i^{(l,m,n)} - 1)$.

- Unimolecular reactions $\mathrm{S}_i^{(l,m,n)} \xrightarrow{c_\mu^{(l,m,n)}}$ Products: $a_\mu^{(l,m,n)} = n_i^{(l,m,n)}\, c_\mu^{(l,m,n)}$ and $\pi_\mu^{(l,m,n);(i)} = c_\mu^{(l,m,n)}$. The "diffusion reactions" representing the jumps from a subvolume to one of its neighbors fall into this category.

- Source reactions $\emptyset \xrightarrow{c_\mu^{(l,m,n)}}$ Products: $a_\mu^{(l,m,n)} = c_\mu^{(l,m,n)}$ and $\pi_\mu^{(l,m,n);(0)} = c_\mu^{(l,m,n)}$.

We use the specific probability rates given in Eq. 7.6. In cases where the 3D computational domain is cubic with equal numbers of subvolumes in each dimension, we use the discretization-corrected specific probability rates as given by Erban and Chapman (2009) [133] for the bimolecular reactions. The computational cost and the formalism of PSRD, however, are independent of the formulation used for the specific probability rates.

**Sampling using partial propensities:** Within the selected subvolume we use partial propensity methods to sample the next reaction according to Eq. 7.17. We group the partial propensities of all reactions within each subvolume according to the index of the factored-out reactant. This results in at most $N+1$ groups of size $O(N)$. Every reaction in a subvolume, and its corresponding partial propensity, are then identifiable by two indices: a *group index* and an *element index*. The group index identifies the partial-propensity group to which a reaction belongs and the element index identifies the position of the reaction inside that group. Determining the index of the next reaction is thus done by first sampling its group index and then the element index.
After the selected reaction has fired and the populations of the involved species have been updated according to the reaction stoichiometry, the affected partial propensities are updated using a dependency graph over species (see Eq. 2.35). This dependency graph points to all partial propensities within the subvolume that need to be updated due to the change in population. If the executed reaction was a "diffusion reaction" modeling the jump of a molecules from a subvolume to one of its neighbours, we additionally update the population of that species in the corresponding neighboring subvolume and update the affected partial-propensities in the neighboring subvolume using the respective dependency over species. Since any partial propensity is a function of the population of at most one species, the number of updates is at most $O(N)$. For more details, see Chapter 2.4.

## 7.2.2 Detailed description of the PSRD algorithm

We provide a detailed description of the algorithms and data structures used in PSRD. The workflow of the algorithm is summarized in Table 7.1.

### 7.2.2.1 Data structures

The population of species in each subvolume $(l,m,n)$ is stored in an array $\boldsymbol{n}^{(l,m,n)}$. The partial propensities of the reactions within each subvolume $(l,m,n)$ are stored in "partial-propensity structures" $\mathbf{\Pi}^{(l,m,n)} = \left\{\mathbf{\Pi}_i^{(l,m,n)}\right\}_{i=0}^{N}$ as one-dimensional arrays of one-dimensional arrays $\mathbf{\Pi}_i^{(l,m,n)}$. Each array $\mathbf{\Pi}_i^{(l,m,n)}$ contains the partial propensities belonging to group $i$ in subvolume $(l,m,n)$. The partial propensities of source reactions are stored as consecutive entries of the $0^{\text{th}}$ array $\mathbf{\Pi}_0^{(l,m,n)}$. The partial propensities of all reactions in subvolume $(l,m,n)$ that have species $\mathrm{S}_1^{(l,m,n)}$ as the factored-out reactant are stored as consecutive entries of $\mathbf{\Pi}_1^{(l,m,n)}$. In general, the $i^{\text{th}}$ array $\mathbf{\Pi}_i^{(l,m,n)}$ contains the partial propensities of all reactions in subvolume $(l,m,n)$ that have $\mathrm{S}_i^{(l,m,n)}$ as the common factored-out reactant, provided these reactions have not yet been included in any of the previous $\mathbf{\Pi}_{j<i}^{(l,m,n)}$. That is, out of the two partial propensities of a bimolecular reaction with $\mathrm{S}_i^{(l,m,n)}$ and $\mathrm{S}_j^{(l,m,n)}$ as its reactants and $i<j$, $\pi_\mu^{(l,m,n);\,(i)}$ is part of $\mathbf{\Pi}_i^{(l,m,n)}$, and $\pi_\mu^{(l,m,n);\,(j)}$ is not stored anywhere. In order to save memory, we lump the "diffusion reactions" of each species within a subvolume into one reaction with no products. The specific probability rate of the lumped reaction

is the sum of the specific probability rates of all "diffusion reactions" in that subvolume. Therefore, instead of storing 6 partial propensities in 3D (4 in 2D), we only store 1 partial propensity for the "diffusion reactions" of each species. This reduces the total number of reactions per subvolume from $M + 6N$ in 3D ($M + 4N$ in 2D) to $M + N$. For convenience, we define all reactions $\mu \leq M$ as "real" reactions and the reaction with index $\mu = M + i$ as the lumped "diffusion reaction" of species $\mathrm{S}_i^{(l,m,n)}$. See Sec. 7.2.2.2 for how the direction of a "diffusion jump" is resolved when a lumped "diffusion reaction" has been selected.
The reaction indices of the partial propensities in $\mathbf{\Pi}^{(l,m,n)}$ are stored in a look-up table $\mathbf{L} = \{\mathbf{L}_i\}_{i=0}^{N}$, which is also an array of arrays. For subvolumes containing the same reaction network, we store the look-up table only once. In case the reaction network is the same in all subvolumes, only a single, global look-up table is needed. Subvolumes that host different reaction networks have different look-up tables. The look-up table renders every reaction within each subvolume identifiable by a unique pair of indices, a group index $I$ and an element index $J$, such that the partial propensity of reaction $\mu = \mathrm{L}_{I,J}$ is stored in $\Pi_{I,J}^{(l,m,n)}$ for subvolume $(l, m, n)$.
The "group-sum array" $\mathbf{\Lambda}^{(l,m,n)}$ stores the sums of the partial propensities in each group $\mathbf{\Pi}_i^{(l,m,n)}$, i.e. $\Lambda_i^{(l,m,n)} = \sum_j \Pi_{i,j}^{(l,m,n)}$. We also store the total propensity of each group in an array $\mathbf{\Sigma}$, computed as $\Sigma_i^{(l,m,n)} = n_i^{(l,m,n)} \Lambda_i^{(l,m,n)}$, $i = 1, \ldots, N$, and $\Sigma_0^{(l,m,n)} = \Lambda_0^{(l,m,n)}$. The total propensity of the reactions in subvolume $(l, m, n)$ is then $a^{(l,m,n)} = \sum_{i=0}^{N} \Sigma_i^{(l,m,n)}$. The total propensity of all reactions across all subvolumes is stored in $a = \sum_{l=1}^{K_x} \sum_{m=1}^{K_y} \sum_{n=1}^{K_z} a^{(l,m,n)}$ and is used to calculate the time to the next reaction according to Eq. 7.15 as $\tau = -a^{-1} \log_2 r_0$, where $r_0$ is a uniformly distributed random number in $[0, 1)$.

#### 7.2.2.2 Algorithms

In PSRD, like in NSM, the subvolume containing the next reaction is sampled first. To this end, the total propensities $a^{(l,m,n)}$ of all subvolumes are sorted into $G_\mathrm{a} = \log_2(a_\mathrm{max}/a_\mathrm{min}) + 1$ bins, such that bin $b$ contains all $a^{(l,m,n)}$ in the interval $2^{b-1} a_\mathrm{min} \leq a^{(l,m,n)} < 2^b a_\mathrm{min}$. The bounds $a_\mathrm{min}$ and $a_\mathrm{max}$ are the smallest non-zero and the largest value that any of the $a^{(l,m,n)}$ can assume during the simulation. They are determined as follows: The lower bound $a_\mathrm{min}$ is the minimum propensity of any reaction in any subvolume when the number of molecules of all reactants is one (minimum non-zero population). For elementary reactions, this is the smallest specific probability rate across all subvolumes. The largest possible value of $a^{(l,m,n)}$ may be ascertained using physical reasoning or prior knowledge about the reaction-diffusion system. In cases where this cannot be evaluated *a priori*, PSRD initially sets $a_\mathrm{max}$ to the maximum $a^{(l,m,n)}$. If during the course of the simulation the maximum $a^{(l,m,n)}$ increases, PSRD updates $a_\mathrm{max}$ and $G_\mathrm{a}$, and the corresponding data structures are dynamically enlarged.
PSRD uses composition-rejection sampling to determine the subvolume of the next reaction in two steps: (i) composition step to find the bin $b$ and (ii) rejection step to find $a^{(l,m,n)}$ inside that bin. The composition step uses linear search to determine

$$b = \min \left[ b' \ : \ r_1 a < \sum_{i=1}^{b'} \alpha_i \right] , \tag{7.18}$$

where $r_1$ is a uniform random number in $[0, 1)$ and $\alpha_i$ is the total propensity in bin $i$ computed by summing up the $a^{(l,m,n)}$ in that bin. The rejection step samples the subvolume $(l, m, n)$ among the

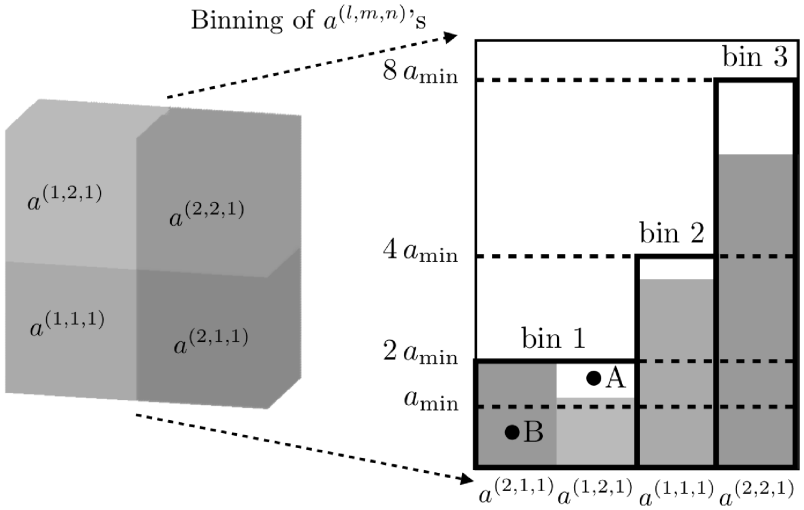



Figure 7.2: Illustration of the binning of the total propensities of the subvolumes used for composition-rejection sampling of the next subvolume. The illustration shows a computational domain divided into 4 subvolumes. Points A and B refer to the example in main text used to explain rejection sampling.

entries in the selected bin $b$. For this, we first generate a uniformly distributed random number $r_2$ in $[0,\, 2^b a_{\min})$ and a uniformly distributed random integer $r_3$ between 1 and the number of entries in bin $b$. If the $r_3$-th element in bin $b$ is greater than or equal to $r_2$, the corresponding subvolume is selected. If the inequality is not satisfied, the rejection step is repeated. This procedure is illustrated in Fig. 7.2 for a computational domain divided into 4 subvolumes with indices $(1,1,1)$, $(2,1,1)$, $(1,2,1)$ and $(2,2,1)$. Assume that the composition step has chosen bin 1 as the bin containing the next subvolume. The rejection step then samples uniformly random points inside the rectangle defining the range of this bin (bold rectangle). A sample is accepted if it falls inside one of the shaded bars representing the $a^{(l,m,n)}$'s. If the first sample (point A in Fig. 7.2 with $r_3 = 2$ and $r_2 > a^{(1,2,1)}$) is rejected, sampling is repeated until the point falls inside one of the shaded bars (point B in Fig. 7.2 with $r_3 = 1$ and $r_2 < a^{(2,1,1)}$). By binning the $a^{(l,m,n)}$ as described above, it is guaranteed that the area covered by the $a^{(l,m,n)}$ bars in each bin is at least 50% of the bin's total area. This ensures that the expected number of rejection steps required is $\leq 2$. The probability of needing more than $k$ rejection steps is $\leq 2^{-k}$ and hence exponentially small.

Once the subvolume $(l,m,n)$ containing the next reaction has been chosen, PSRD samples the index of the next reaction $\mu$ within that subvolume in two steps: (i) perform linear search for the group index $I$ such that

$$I = \min\left[ I' \;:\; r_4 a^{(l,m,n)} < \sum_{i=0}^{I'} \Sigma_i^{(l,m,n)} \right] \tag{7.19}$$

and (ii) perform linear search for the element index $J$ inside group $\mathbf{\Pi}_I^{(l,m,n)}$ such that

$$J = \min\left[J' \;:\; r_4 a^{(l,m,n)} < \sum_{j=1}^{J'} n_I^{(l,m,n)} \Pi_{I,j}^{(l,m,n)} + \left(\sum_{i=0}^{I} \Sigma_i^{(l,m,n)}\right) - \Sigma_I^{(l,m,n)}\right], \tag{7.20}$$

where $r_4$ is a uniform random number in [0,1) (see Eqs. 2.33 and 2.34 for the procedure to sample $J$ efficiently). The indices $I$ and $J$ are then translated to the reaction index $\mu$ in subvolume $(l,m,n)$ using the look-up table $\mathbf{L}$, thus $\mu = \mathrm{L}_{I,J}$. In order to reduce the average search depth, the group and element indices are dynamically rearranged such that frequent reactions accumulate at the beginning of the list, i.e., have low index values. This is done by dynamically bubbling up a reaction whenever it fires by performing a single iteration of a bubble-sort algorithm. The permutation lists for the reordered indices in each subvolume are stored in an array for the $I$'s, and one-dimensional array of one-dimensional arrays of the size of $\mathbf{\Pi}^{(l,m,n)}$ for the $J$'s. PSRD thus uses the sorting partial-propensity direct method (SPDM) to sample the next reaction within a subvolume (see Sec. 2.4.4). This renders the sampling procedure more efficient (in the sense that it reduces the prefactor in the scaling of the computational cost) when the reaction network in a subvolume is multi-scale (stiff), without compromising on the efficiency in non-stiff cases. In SRD simulations the reaction networks inside the subvolumes tend to be stiff since the specific probability rates of bimolecular reactions scale as $h^{-3}$ (in 3D subvolumes) whereas those of source reactions scale as $h^3$ (see Eq. 7.6). Using SPDM instead of PDM may hence lead to significant computational savings.

Once a reaction has been executed, $\boldsymbol{n}^{(l,m,n)}$, $\mathbf{\Pi}^{(l,m,n)}$, $\mathbf{\Lambda}^{(l,m,n)}$, and $\mathbf{\Sigma}^{(l,m,n)}$ need to be updated. This is efficiently done using three update structures. If the reaction network is the same in each subvolume, the same update structures can be used for all subvolumes and they do not have to be stored separately for different subvolumes. Subvolumes containing different reaction networks have different update structures.

- $\mathbf{U}^{(1)}$ is an array of $M$ arrays, where the $i^{\text{th}}$ array contains the indices of all species involved in the $i^{\text{th}}$ "real" reaction. The index of the species involved in the $i^{\text{th}}$ lumped "diffusion reaction" does not need to be stored as it is simply $i$ itself.

- $\mathbf{U}^{(2)}$ is a array of $M$ arrays containing the corresponding stoichiometries (the change in population of each species upon reaction) of the species stored in $\mathbf{U}^{(1)}$. The stoichiometries of the "diffusion reactions" are not stored since they are all $-1$.

- $\mathbf{U}^{(3)}$ is a array of $N$ arrays, where the $i^{\text{th}}$ array contains the indices of all entries in the $\mathbf{\Pi}^{(l,m,n)}$'s that depend on $n_i^{(l,m,n)}$.

When a reaction is executed in subvolume $(l,m,n)$, the populations of the species involved in this reaction change. Hence, all entries in $\mathbf{\Pi}^{(l,m,n)}$ that depend on these populations need to be updated. After each reaction, we use $\mathbf{U}^{(1)}$ to determine the indices of all species involved in this reaction. The stoichiometry is then looked up in $\mathbf{U}^{(2)}$ and the population $\boldsymbol{n}^{(l,m,n)}$ is updated. Subsequently, $\mathbf{U}^{(3)}$ is used to locate the affected entries in $\mathbf{\Pi}^{(l,m,n)}$ and recompute them. The two data structures $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ hence amount to a sparse representation of the stoichiometry matrix; $\mathbf{U}^{(3)}$ represents the dependency graph over species. Since the partial propensities of unimolecular and source reactions are constant and never need to be updated, $\mathbf{U}^{(3)}$ only contains the indices of the partial propensities of bimolecular reactions. Along with updating the partial propensities in subvolume $(l,m,n)$, the

change in the total propensity of that subvolume is also calculated and incrementally applied to $a^{(l,m,n)}$. This may require the bin membership of $a^{(l,m,n)}$ to be updated, for which the current bin assignment of $a^{(l,m,n)}$ must be known. We implement this by storing two additional integers for every $a^{(l,m,n)}$: one for the bin membership and the other for the location inside that bin. Depending on its new value, $a^{(l,m,n)}$ may remain in the same bin or move to a different one. Removal of an element from a bin is done by replacing it with the last element in that bin and reducing the bin size by one. Addition of an element into a bin is done by appending it at end of the bin. The computational cost of both of these operations is $O(1)$ [18] (see Sec. 2.4.5).

If the index of the next reaction is greater than $M$, then the sampled reaction is a lumped "diffusion reaction" and additional steps need to be taken to resolve the direction of the jump as follows: First, the index of the species undergoing diffusion is computed as $i = \mu - M$. Second, a uniform random number between 0 and the lumped specific probability rate of the lumped "diffusion reaction" is generated. Third, linear search over the specific probability rates of individual directional diffusion events is used to determine the target subvolume of the jump. The jump is executed by increasing the population of species $\mathrm{S}_i$ in the target subvolume by 1 (the reduction in the source subvolume has already been done above) and updating the entries in the partial propensity structure of the target subvolume as given by the indices in $\mathbf{U}_i^{(3)}$. Finally, the total propensity of the target subvolume and its bin membership are updated.

Figure 7.3 summarizes the data structures used in PSRD for the example reaction network given in Eq. 7.1. The complete algorithm is given in Table 7.1. The computational cost of PSRD to sample the subvolume is $O(1)$ if the ratio of maximum to minimum non-zero total propensity in each subvolume is independent of the number of subvolumes and of the size of the reaction network. In cases where this ratio is not bounded by a constant, the computational cost to sample the subvolume is $O(G_\mathrm{a})$, where the total number of bins $G_\mathrm{a}$ depends on the logarithmic span of the subvolume propensities as $G_\mathrm{a} = \log_2(a_\mathrm{max}/a_\mathrm{min})+1$. The computational cost of sampling the index of the next reaction within a subvolume is $O(N)$. The overall computational cost of PSRD hence is $O(G_\mathrm{a}+N)$, which is at most $O(\log_2 N_\mathrm{v} + N)$. The memory requirement of PSRD is $O((M+N)N_\mathrm{v})$. For more details on the computational cost and the memory requirement, see Sec. 7.2.3.

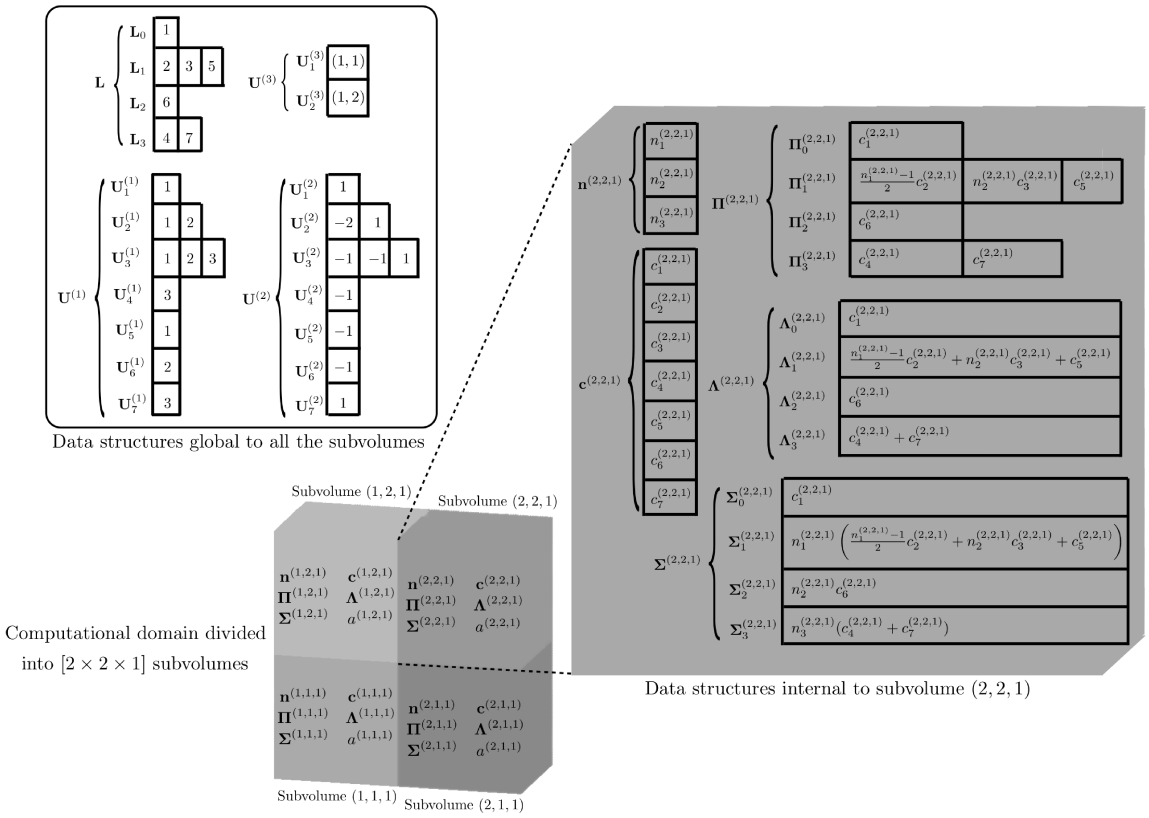



Figure 7.3: The data structures in PSRD. The contents of the data structures shown corresponds to the example reaction network in Eq. 7.1 with 3 species and 4 reactions. We assume that the computational domain is divided into 4 subvolumes. In the illustration, $c_5^{(2,2,1)}$, $c_6^{(2,2,1)}$ ,and $c_7^{(2,2,1)}$ are the lumped specific probability rates of the "diffusion reactions" of species 1, 2, and 3 respectively. See main text for further details.

0. Assume that the reaction network has $N$ species and $M$ reactions. Divide the computational box into $N_\mathrm{v} = K_x K_y K_z$ cubic subvolumes of edge length $h$. Formulate the reaction network modeling the reaction-diffusion system by including the jump reactions. The resulting reaction network for the reaction-diffusion system has at most $M + 6N$ reactions and $N$ species in each subvolume. Lump the "diffusion reactions" of each species in each subvolume into one reaction with no products, such that number of reactions in each subvolume is $M + N$. Make sure that the reaction index of the lumped "diffusion reactions" is $> M$ and that reaction $\mu = M + i$ is the lumped "diffusion reaction" of species $i$.

1. Set time $t \leftarrow 0$. Initialize the data structures in each subvolume $(l, m, n)$: the partial-propensity structure $\mathbf{\Pi}^{(l,m,n)}$, the group-sum array $\mathbf{\Lambda}^{(l,m,n)}$, $\mathbf{\Sigma}^{(l,m,n)}$, the population $\mathbf{n}^{(l,m,n)}$, the specific probability rates $\mathbf{c}^{(l,m,n)}$, and the total propensity in the subvolume $a^{(l,m,n)}$. Also initialize the data structures global to all subvolumes: the look-up table $\mathbf{L}$, the sparse representation of the stoichiometry matrix $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$, the dependency graph over species $\mathbf{U}^{(3)}$, and the total propensity of all subvolumes $a$. Bin the $a^{(l,m,n)}$ into $G_\mathrm{a}$ bins as described in Sec. 7.2.2.2.

2. While $t < t_\mathrm{f}$, repeat:

   2.1. Compute the time to the next reaction $\tau \leftarrow a^{-1} \ln(r^{-1})$, where $a$ is the total propensity of all reactions and $r$ a uniformly distributed random number in $[0, 1)$.

   2.2. Use composition-rejection sampling to determine the subvolume $(l, m, n)$ containing the next reaction. Use linear search (Eq. 7.18) in the composition step to locate the bin containing $a^{(l,m,n)}$ and use the rejection step to locate $a^{(l,m,n)}$ inside that bin.

   2.3. Sample the next reaction $\mu$ in subvolume $(l, m, n)$ by sampling its group and element indices. Sample the group index $I$ using linear search over $\mathbf{\Sigma}^{(l,m,n)}$ (Eq. 7.19). Subsequently, sample the element index $J$ using linear search over $\mathbf{\Pi}_I^{(l,m,n)}$ (Eq. 7.20). The reaction index $\mu$ is then identified from the look-up table as $\mu = \mathrm{L}_{I,J}$.

   2.4. Update the internal data structures in subvolume $(l, m, n)$ and the total propensity $a$ using $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$.

   2.5. Increase the number of bins $G_\mathrm{a}$ if necessary and update the bin membership of $a^{(l,m,n)}$

   2.6. If $\mu > M$ (i.e., the sample reaction is a lumped "diffusion reaction"), compute the index of the diffusing species as $s = \mu - M$. Resolve the diffusion event to identify the neighboring target subvolume $(l', m', n')$ to which species $s$ of subvolume $(l, m, n)$ is diffusing. Update the population of species $s$ in the target subvolume. Subsequently, update the other internal data structures of the target subvolume and the total propensity $a$ using $\mathbf{U}_s^{(3)}$. Increase the number of bins $G_\mathrm{a}$ if necessary and update the bin membership of $a^{(l',m',n')}$.

   2.6. Advance time: $t \leftarrow t + \tau$.

3. Stop.

Table 7.1: The detailed algorithm of PSRD.

### 7.2.3 Computational cost

The steps that define the scaling of the computational cost of PSRD are the sampling of the subvolume containing the next reaction, the sampling of the next reaction within that subvolume, and the update of the data structures after firing the sampled reaction.

The computational cost of the composition-rejection sampling of the next subvolume is $O(G_\mathrm{a})$. This is because (i) the composition step is a linear search over $G_\mathrm{a}$ bins, and (ii) the rejection step is $O(1)$ since the average number of iterations for this step is bounded by a constant thanks to the dyadic binning [18] (see Sec. 2.4.5).

The computational cost for sampling the next reaction within the selected subvolume $(l, m, n)$ is $O(N)$. This step involves sampling the group index $I$ and the element index $J$ of the next reaction in the partial-propensity structure. Sampling the group index involves a linear search over the at most $N+1$ elements of $\mathbf{\Sigma}^{(l,m,n)}$ and hence has a computational cost of $O(N)$. Sampling the element index involves a linear search over the $O(N)$ elements of $\mathbf{\Pi}_I^{(l,m,n)}$ and hence has a computational cost of $O(N)$ as well.

The computational cost for updating the data structures within a subvolume is at most $O(N)$. Assuming that the number of distinct species involved in any one chemical reaction is $O(1)$ (i.e., does not increase beyond a constant bound as the number of species in the network increases), the cost of updating the population of species is $O(1)$. Under the same assumption, the number of entries in $\mathbf{\Pi}^{(l,m,n)}$ that need to be updated after any reaction has fired is at most $O(N)$.

By the same argument, the cost of updating the partial-propensity structure of any neighboring subvolume upon firing of a "diffusion reaction" is at most $O(N)$.

Overall, the computational cost of PSRD thus is $O(G_\mathrm{a} + N)$, irrespective of the fraction $f_\mathrm{r}$ of "real" reaction firings.

The asymptotically (for large $N_\mathrm{v}$) worst case for PSRD is when half of the subvolumes contain bimolecular reactions and the other half source reactions. In 3D subvolumes, the propensity of the bimolecular reactions is proportional to $h^{-3}$ whereas that of the source reactions is proportional to $h^3$, where $h$ is the edge length of the subvolumes. As $N_\mathrm{v}$ increases, the logarithmic span of the propensities in the system hence increases. This leads to an increase in the number of bins $G_\mathrm{a}$ that is proportional to $\log_2 h^{-6} = 2\log_2 N_\mathrm{v} - 2\log_2 \Omega$, where $\Omega$ is the (constant) volume of the reactor. Therefore, $G_\mathrm{a} \in O(\log_2 N_\mathrm{v})$, rendering the computational cost of PSRD $O(\log_2 N_\mathrm{v} + N)$ in the worst case, independent of $f_\mathrm{r}$. This worst-case scaling of PSRD's computational cost can be reduced to $O(\log_2 \log_2 N_\mathrm{v} + N)$ by using a tree search [1] to sample $b$ in Eq. 7.18.

The data structures of PSRD require $O(M + N)$ memory per subvolume. Therefore, the total memory requirement of PSRD is $O((M + N)N_\mathrm{v})$.

### 7.2.4 Benchmarks

We analyze the computational cost of PSRD as quantified by the average simulation (CPU) time $\Theta$ taken per reaction event of the chemical reaction model of a reaction-diffusion system. We compare it to the time expected from the theoretical cost analysis (see Sec. 7.2.3) for two different types of reaction networks: (i) an aggregation model where the number of reactions increases super-linearly with the number of species and (ii) a linear chain model where the number of reactions is almost the same as the number of species. We simulate the corresponding reaction-diffusion processes in a three-dimensional cubic computational domain (reactor) of dimensions $L \times L \times L$ and volume $\Omega = L^3$ from a initial time $t = 0$ until a final time $t = t_\mathrm{f}$. For simulating the reaction-diffusion

process we divide the computational domain into $N_\mathrm{v} = K^3$ equi-sized cubic subvolumes of edge length $h = L/K$, such that $K$ is the number of subvolumes along each spatial dimension.

For each of these networks we report $\Theta$ as a function of the number of subvolumes $N_\mathrm{v}$ for a fixed size of the reaction network and as a function of the reaction network size for a fixed number of subvolumes. We use the number of species $N$ in the reaction network to quantify the size of the network. All timings are compared to those obtained on the same systems and the same computer using NSM.

Both PSRD and NSM were implemented in C++ using the random number generator of the GSL library and compiled using the Intel C++ compiler version 12.0.2 with the O3 optimization flag. NSM is implemented according to the details provided on the MesoRD webpage (Algorithm 7 in [140]). All timings were measured on a Linux 2.6 workstation with a 2.8 GHz quad-core Intel Xeon E5462 processor, 8 GB of memory and 4 MB L2 cache. For all test cases, we simulate until a final time $t_\mathrm{f}$ and report the average CPU time $\Theta$ per reaction event. The time $\Theta$ does not include the initialization of the data structures as this is done only once. We explain the measurements by least-squares fits of $\Theta(N_\mathrm{v}, N)$ with the corresponding theoretical cost models. For PSRD and NSM, we hence fit $\Theta$ with $\alpha_1 \log_2 N_\mathrm{v} + \alpha_2 N$ and $\alpha_1 \log_2 N_\mathrm{v} + \alpha_2 f_\mathrm{r} M + \alpha_3 (1 - f_\mathrm{r}) 6N$, respectively. Before fitting, we estimate the functional dependence of $f_\mathrm{r}$ on $N_\mathrm{v}$ or $N$ by performing simulations. Subsequently, we fit $\Theta$ to determine the coefficients $\alpha_i$.

All simulations are run without any *a priori* estimate of the maximum total propensity $a_\mathrm{max}$ across all subvolumes. Instead, $a_\mathrm{max}$ is constantly updated during a simulation and the number of bins $G_\mathrm{a}$ is dynamically increased when required (see Sec. 7.2.2.2).

#### 7.2.4.1 Colloidal aggregation model

We consider the nonequilibrium colloidal aggregation model as a prototype of a strongly coupled reaction network in which the number of reactions increases super-linearly with the number of species:

$$\begin{array}{ccccc} \emptyset & \xrightarrow{k_0} & \mathrm{S}_1 & & \\ \mathrm{S}_i + \mathrm{S}_j & \xrightarrow{k_{i,j}} & \mathrm{S}_{i+j} & i + j = 2, \ldots, N \\ \mathrm{S}_{i+j} & \xrightarrow{\bar{k}_{i,j}} & \mathrm{S}_i + \mathrm{S}_j & i + j = 2, \ldots, N \\ \mathrm{S}_i & \xrightarrow{k_i} & \emptyset & i = 1, \ldots, N. \end{array} \tag{7.21}$$

The $k$'s are the macroscopic reaction rates. This system of reactions models the influx of monomers ($\mathrm{S}_1$) into a reactor where multimers ($\mathrm{S}_i$) fuse with each other to form larger multimers. Multimers in the reactor also break to form smaller units in all possible combinations, and all of the multimers can leave the reactor. Such a system of reactions models driven colloidal aggregation and is relevant for a variety of phenomena of practical importance, e.g., nano-particle clustering and colloidal crystallization (nanotechnology), emulsification and emulsion stabilization in porous media (oil industry), and oligomerization of proteins (biochemistry). For $N$ chemical species, the aggregation reaction network consists of $M = \lfloor N^2/2 \rfloor + N + 1$ reactions.

We divide the cubic computational domain (reactor) into $N_\mathrm{v} = K^3$ subvolumes, such that the on-lattice reaction-diffusion process with reflective boundary conditions is described by the following

set of "reactions":

$$\left.\begin{array}{ccccc} \emptyset & \xrightarrow{k_0} & \mathrm{S}_1^{(l,m,n)} & & \\ \mathrm{S}_i^{(l,m,n)} + \mathrm{S}_j^{(l,m,n)} & \xrightarrow{k_{i,j}} & \mathrm{S}_{i+j}^{(l,m,n)} & i+j=2,\ldots,N \\ \mathrm{S}_{i+j}^{(l,m,n)} & \xrightarrow{\bar{k}_{i,j}} & \mathrm{S}_i^{(l,m,n)} + \mathrm{S}_j^{(l,m,n)} & i+j=2,\ldots,N \\ \mathrm{S}_i^{(l,m,n)} & \xrightarrow{k_i} & \emptyset & i=1,\ldots,N \\ \mathrm{S}_i^{(l,m,n)} & \xrightarrow{D_i/h^2} & \mathrm{S}_i^{\boldsymbol{c}} & i=1,\ldots,N \end{array}\right\} \quad \forall\ (l,m,n) \in I_{\mathrm{all}},\ \boldsymbol{c} \in C_{l,m,n}, \tag{7.22}$$

where $D_i$ is the diffusion constant of species $\mathrm{S}_i$ and $h$ the edge length of the cubic subvolumes. The propensities of these reactions are computed as described in Sec. 7.1. For all bimolecular reactions, we use discretization-corrected propensities (see Eqs. 7.9 and 7.11). The above network consists of $NK^3 = NN_\mathrm{v}$ species and $MK^3 + N(6K^3 - 6K^2) = (\lfloor N^2/2 \rfloor + 1)N_\mathrm{v} + N(7N_\mathrm{v} - 6N_\mathrm{v}^{2/3})$ reactions. For the present benchmarks, we set the macroscopic reaction rates $k_{i,i} = 0.5$, all other rates and all diffusion constants to 1, and the reactor volume to $\Omega = 10$. At time $t = 0$, the populations of all species in all subvolumes, i.e. all $n_i^{(l,m,n)}$, are set to 0. From this initial condition we simulate the reaction-diffusion system until $t_\mathrm{f} = 100$.

Figure 7.4A shows the computational cost $\Theta$ as a function of the number of subvolumes $N_\mathrm{v}$ using PSRD and NSM for two fixed-size aggregation networks with $N = 10$ and $N = 100$, respectively. The corresponding numbers of reactions $M$ are 61 and 5101, respectively. In both cases we estimate $f_\mathrm{r}$ and use it for fitting $\Theta$. We observe that $f_\mathrm{r}$ decreases as $N_\mathrm{v}^{-0.34}$ with increasing $N_\mathrm{v}$. For PSRD, $\Theta(N_\mathrm{v}, N = 10) \approx 0.02861 \log N_\mathrm{v}$ at large $N_\mathrm{v}$. This scaling of $\Theta$ is caused by the dynamic increase in the number of bins $G_\mathrm{a}$. For NSM, $\Theta(N_\mathrm{v}, N = 10) \approx 0.1095 \log N_\mathrm{v}$ at large $N_\mathrm{v}$. For the larger network with $N = 100$, we find for PSRD $\Theta(N_\mathrm{v}, N = 100) \approx 0.04401 \log N_\mathrm{v}$ at large $N_\mathrm{v}$. For NSM, $\Theta(N_\mathrm{v}, N = 100) \approx 0.288 \log N_\mathrm{v}$ at large $N_\mathrm{v}$. For smaller $1 \leq N_\mathrm{v} < 512$, $\Theta$ of NSM decreases with increasing $N_\mathrm{v}$. This is mediated by the decrease in $f_\mathrm{r}$. At $N_\mathrm{v} = 1$, $f_\mathrm{r} = 1$ and the cost is dominated by that of sampling the next "real" reaction. As $N_\mathrm{v}$ increases, $f_\mathrm{r}$ decreases. This decrease in $f_\mathrm{r}$ progressively reduces the cost of sampling a reaction in a subvolume from being linear in $M$ to linear in $6N$. At large-enough $N_\mathrm{v}$, the cost of sampling a reaction in a subvolume is dominated by the cost of sampling "diffusion reactions". For a fixed network size, the increase in $\Theta(N_\mathrm{v}, N = 100)$ at large $N_\mathrm{v}$ is thus primarily due to the increasing cost to sample the next subvolume. In summary, the scaling of the computational cost of PSRD with respect to the number of subvolumes $N_\mathrm{v}$ is $O(\log_2 N_\mathrm{v})$. This scaling is asymptotically (for large $N_\mathrm{v}$) the same as that of NSM, but with a smaller prefactor.

Figure 7.4B shows the computational cost $\Theta$ as a function of the size $N$ of the aggregation reaction network using PSRD and NSM with $N_\mathrm{v} = 512$ and $N_\mathrm{v} = 1000$ subvolumes. We observe that for both $N_\mathrm{v}$ the ratio $f_\mathrm{r}$ does not depend on the size $N$ of network. For $N_\mathrm{v} = 512$, $f_\mathrm{r} = 0.04$, decreasing to $f_\mathrm{r} = 0.02$ for $N_\mathrm{v} = 1000$. For PSRD, $\Theta(N_\mathrm{v} = 512, N) \approx 0.002258\, N$, confirming the linear dependence on $N$ predicted by the theoretical cost analysis. For NSM, $\Theta(N_\mathrm{v} = 1000, N) \approx 0.00011M + 0.0152N$. For the larger number subvolumes, $N_\mathrm{v} = 1000$, $\Theta(N_\mathrm{v} = 1000, N) \approx 0.002777N$ for PSRD. For NSM, $\Theta(N_\mathrm{v} = 1000, N) \approx 0.000055M + 0.0186N$. In summary, the scaling of the computational cost of PSRD with respect to the size $N$ of the reaction network is $O(N)$.

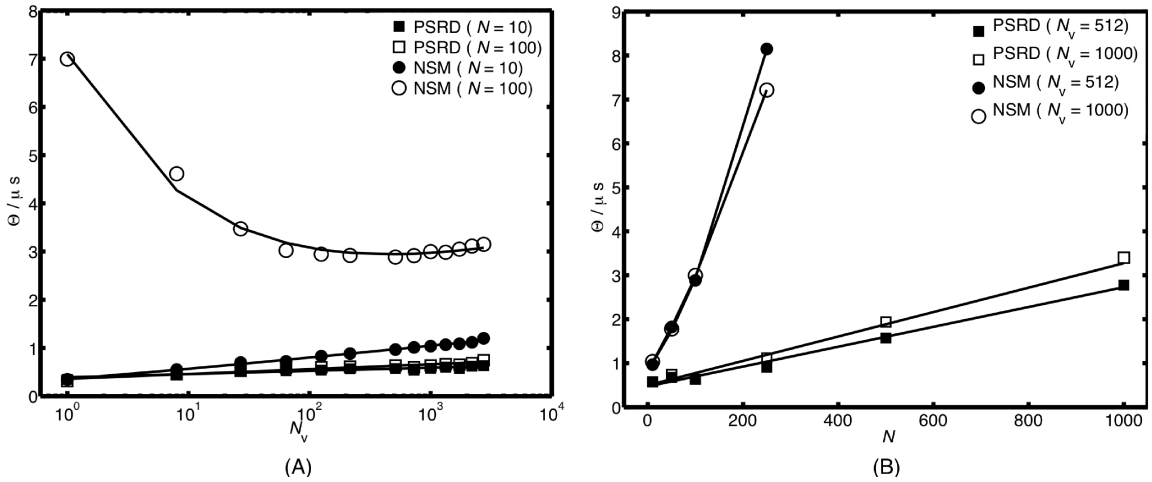



Figure 7.4: Computational cost of PSRD and NSM for the aggregation model (Eq. 7.21). (A) Computational cost $\Theta$ of PSRD (squares) and NSM (circles) as a function of the number of subvolumes $N_\mathrm{v}$ with the size of the reaction network fixed to $N = 10$ (filled symbols) and $N = 100$ (empty symbols), respectively. The solid lines show the corresponding least-squares fits of the theoretical cost models: For $N = 10$, $\Theta^{\mathrm{PSRD}} \approx 0.02861 \log N_\mathrm{v} + 0.03925N$, $\Theta^{\mathrm{NSM}} \approx 0.1095 \log N_\mathrm{v} + 0.00581 f_\mathrm{r} M + 0.00481(1 - f_\mathrm{r})6N$; for $N = 100$, $\Theta^{\mathrm{PSRD}} \approx 0.04401 \log N_\mathrm{v} + 0.003579N$, $\Theta^{\mathrm{NSM}} \approx 0.288 \log N_\mathrm{v} + 0.001375 f_\mathrm{r} M + 0.001418(1 - f_\mathrm{r})6N$. We estimate $f_\mathrm{r} \approx 1.096 N_\mathrm{v}^{-0.3353} - 0.08263$ for $N = 10$ and $f_\mathrm{r} = 1.097 N_\mathrm{v}^{-0.3372} - 0.0825$ for $N = 100$. (B) Computational cost $\Theta$ of PSRD (squares) and NSM (circles) as a function of the number of species $N$ in the reaction network with the number of subvolumes fixed to $N_\mathrm{v} = 512$ (filled symbols) and $N_\mathrm{v} = 1000$ (empty symbols), respectively. The solid lines show the corresponding least-squares fits of the theoretical cost models: For $N_\mathrm{v} = 512$, $\Theta^{\mathrm{PSRD}} \approx 0.07559 \log N_\mathrm{v} + 0.002258N$, $\Theta^{\mathrm{NSM}} \approx 0.1356 \log N_\mathrm{v} + 0.002784 f_\mathrm{r}(\lfloor N^2/2 \rfloor + N + 1) + 0.002633(1 - f_\mathrm{r})6N$; for $N_\mathrm{v} = 1000$, $\Theta^{\mathrm{PSRD}} \approx 0.07205 \log N_\mathrm{v} + 0.002777N$, $\Theta^{\mathrm{NSM}} \approx 0.1198 \log N_\mathrm{v} + 0.002762 f_\mathrm{r}(\lfloor N^2/2 \rfloor + N + 1) + 0.003163(1 - f_\mathrm{r})6N$. The fraction $f_\mathrm{r} = 0.04$ for $N_\mathrm{v} = 512$ and $f_\mathrm{r} = 0.02$ for $N_\mathrm{v} = 1000$.

#### 7.2.4.2 Linear chain model

As a prototypical reaction network in which the number of reactions is almost the same as the number of species, we consider the nonequilibrium linear chain model:

$$\begin{array}{lcll} \emptyset & \xrightarrow{k_0} & \mathrm{S}_1 & \\ \mathrm{S}_i & \xrightarrow{k_i} & \mathrm{S}_{i+1} & i = 1, \ldots, N-1 \\ \mathrm{S}_N & \xrightarrow{k_N} & \emptyset. & \end{array} \tag{7.23}$$

Again, the $k$'s are the macroscopic reaction rates. This linear chain of reactions can, e.g., be used to model signal transduction pathways in biological cells [46, 40]. For $N$ species, this network contains $M = N + 1$ reactions.

Again dividing the cubic computational domain into $N_\mathrm{v} = K^3$ subvolumes, the resulting reaction-

diffusion system with reflective boundary conditions is given by:

$$\left.\begin{array}{lcll} \emptyset & \xrightarrow{k_0} & \mathrm{S}_1^{(l,m,n)} & \\ \mathrm{S}_i^{(l,m,n)} & \xrightarrow{k_i} & \mathrm{S}_{i+1}^{(l,m,n)} & i = 1,\ldots,N-1 \\ \mathrm{S}_N^{(l,m,n)} & \xrightarrow{k_N} & \emptyset & \\ \mathrm{S}_i^{(l,m,n)} & \xrightarrow{D_i/h^2} & \mathrm{S}_i^{\boldsymbol{c}} & i = 1,\ldots,N \end{array}\right\} \quad \forall\ (l,m,n) \in I_{\mathrm{all}},\ \boldsymbol{c} \in C_{l,m,n}, \tag{7.24}$$

where $D_i$ is the diffusion constant of species $\mathrm{S}_i$ and $h$ is the edge length of the cubic subvolumes. The propensities of these reactions are computed as described in Sec. 7.1. There are no bimolecular reactions in this network, and we do not use discretization-corrected propensities [133]. This system of reactions modeling the reaction-diffusion process contains $NK^3 = NN_\mathrm{v}$ species and $MK^3 + N(6K^3 - 6K^2) = (7N+1)N_\mathrm{v} - 6NN_\mathrm{v}^{2/3}$ reactions.

For the benchmarks we set all macroscopic reactions rates and all diffusion constants $D_i$ to 1, and the volume of the reactor to $\Omega = 100$. At time $t = 0$, the populations of all species in all subvolumes are 0, and the simulation is run until $t_\mathrm{f} = 100$.

Figure 7.5A shows the computational cost $\Theta$ as a function of the number of subvolumes $N_\mathrm{v}$ using PSRD and NSM for two fixed-size linear chain networks with $N = 10$ and $N = 100$. The corresponding numbers of reactions $M$ are 11 and 101, respectively. In both cases we estimate $f_\mathrm{r}$ and use it for fitting $\Theta$. We observe that $f_\mathrm{r}$ decreases as $N_\mathrm{v}^{-0.22}$ with increasing $N_\mathrm{v}$. For PSRD, $\Theta(N_\mathrm{v}, N = 10) \approx 0.03312 \log N_\mathrm{v}$. This scaling of $\Theta$ is caused by the increase in the number of bins $G_\mathrm{a}$. For NSM, $\Theta(N_\mathrm{v}, N = 10) \approx 0.08256 \log N_\mathrm{v}$. For the larger network with $N = 100$, the computational cost of PSRD is $\Theta(N_\mathrm{v}, N = 100) \approx 0.04842 \log N_\mathrm{v}$ for $N_\mathrm{v} \lessapprox 512$ and $\Theta(N_\mathrm{v}, N = 100) \approx 0.2923 \log N_\mathrm{v}$ for $N_\mathrm{v} \gtrapprox 512$. For NSM, $\Theta(N_\mathrm{v}, N = 100) \approx 0.1428 \log N_\mathrm{v}$ for $N_\mathrm{v} \lessapprox 512$ and $\Theta(N_\mathrm{v}, N = 100) \approx 0.5929 \log N_\mathrm{v}$ for $N_\mathrm{v} \gtrapprox 512$. The abrupt increase in the prefactor of the scaling around $N_\mathrm{v} \approx 512$ is likely caused by cache-memory effects. In summary, the scaling of the computational cost of PSRD with respect to the number of subvolumes $N_\mathrm{v}$ is $O(\log_2 N_\mathrm{v})$. Again, this is the same scaling as that of NSM, but with a smaller prefactor.

Figure 7.5B shows the computational cost $\Theta$ as a function of the size $N$ of the linear chain network using PSRD and NSM with $N_\mathrm{v} = 512$ and $N_\mathrm{v} = 1728$ subvolumes. We observe that for both $N_\mathrm{v}$ the ratio $f_\mathrm{r}$ is independent of the size $N$ of the network. For $N_\mathrm{v} = 512$, $f_\mathrm{r} = 0.06$, decreasing to $f_\mathrm{r} = 0.03$ for $N_\mathrm{v} = 1728$. We observe that the scaling of $\Theta$ is slower than predicted by the theoretical cost analysis. This is not a violation of the theory since the theoretical analysis only provides an upper bound for the scaling. The slower scaling in the present case is specific to the particular reaction network. We obtain reasonable fits with a function linear in $\log N$. The asymptotic plateau is due to "diffusion reactions" of species $\mathrm{S}_1$ accounting for the majority of all reaction firings. Since this reaction is on top of the list (species index 1), it is found in $O(1)$ time. For PSRD, $\Theta(N_\mathrm{v} = 512, N) \approx 0.03051 \log N$. For NSM, $\Theta(N_\mathrm{v} = 512, N) \approx 0.07885 \log N$. For the larger number subvolumes $N_\mathrm{v} = 1728$, $\Theta$ of PSRD is $\Theta(N_\mathrm{v} = 1728, N) \approx 0.08479 \log N$. For NSM, $\Theta(N_\mathrm{v} = 1728, N) \approx 0.1642 \log N$. In summary, the scaling of the computational cost of PSRD with respect to the size $N$ of the reaction network is $O(N)$, since $\log N \in O(N)$.

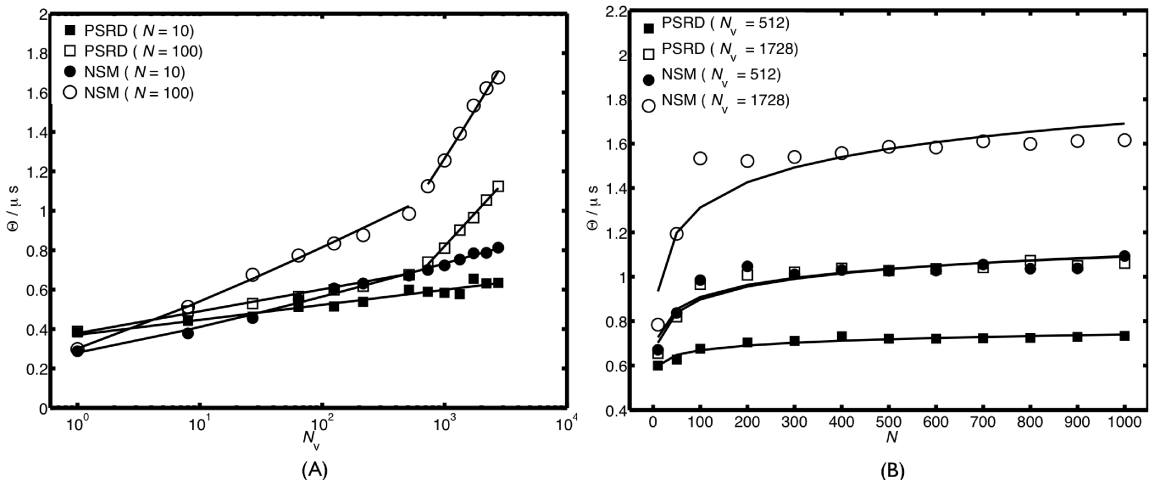



Figure 7.5: Computational cost of PSRD and NSM for the linear chain model (Eq. 7.23). (A) Computational cost $\Theta$ of PSRD (squares) and NSM (circles) as a function of the number of subvolumes $N_\mathrm{v}$ with the size of the reaction network fixed to $N = 10$ (filled symbols) and $N = 100$ (empty symbols), respectively. The solid lines show the corresponding least-squares fits of the theoretical cost models: For $N = 10$, $\Theta^\mathrm{PSRD} \approx 0.03312 \log N_\mathrm{v} + 0.03703N$, $\Theta^\mathrm{NSM} \approx 0.08256 \log N_\mathrm{v} + 0.02504 f_\mathrm{r} M + 0.002615(1 - f_\mathrm{r})6N$; for $N = 100$, $\Theta^\mathrm{PSRD} \approx 0.04842 \log N_\mathrm{v} + 0.003786N$, $\Theta^\mathrm{NSM} \approx 0.1428 \log N_\mathrm{v} + 0.002934 f_\mathrm{r} M + 0.0001978(1 - f_\mathrm{r})6N$ for $N_\mathrm{v} \lessapprox 512$ and $\Theta^\mathrm{PSRD} \approx 0.2923 \log N_\mathrm{v} - 0.01199N$, $\Theta^\mathrm{NSM} \approx 0.5929 \log N_\mathrm{v} + 0.0000008 f_\mathrm{r} M - 0.004924(1 - f_\mathrm{r})6N$ for $N_\mathrm{v} \gtrapprox 512$. We estimate $f_\mathrm{r} \approx 1.281 N_\mathrm{v}^{-0.2121} - 0.2505$ for $N = 10$ and $f_\mathrm{r} \approx 1.241 N_\mathrm{v}^{-0.2291} - 0.2138$ for $N = 100$. (B) Computational cost $\Theta$ of PSRD (squares) and NSM (circles) as a function of the number of species $N$ in the reaction network with the number of subvolumes fixed to $N_\mathrm{v} = 512$ (filled symbols) and $N_\mathrm{v} = 1728$ (empty symbols), respectively. The solid lines show the corresponding least-squares fits of the theoretical cost models: For $N_\mathrm{v} = 512$, $\Theta^\mathrm{PSRD} \approx 0.03051 \log N + 0.5291$, $\Theta^\mathrm{NSM} \approx 0.07885 \log N + 0.5458$; for $N_\mathrm{v} = 1728$, $\Theta^\mathrm{PSRD} \approx 0.08479 \log N + 0.5073$, $\Theta^\mathrm{NSM} \approx 0.1642 \log N + 0.5561$. The fraction $f_\mathrm{r} = 0.06$ for $N_\mathrm{v} = 512$ and $f_\mathrm{r} = 0.03$ for $N_\mathrm{v} = 1728$.

### 7.2.5 Two- and three-dimensional SRD simulations using PSRD

As an example application we use PSRD for two- and three-dimensional SRD simulations of the Gray-Scott reaction system [128, 129, 130, 98, 131], given by:

$$\begin{aligned}
\varnothing &\xrightarrow{Fk_1u^3} \mathrm{S}_1 \\
\mathrm{S}_1 &\xrightarrow{Fk_1u^2} \varnothing \\
\mathrm{S}_1 + 2\mathrm{S}_2 &\xrightarrow{k_1} 3\mathrm{S}_2 \\
\mathrm{S}_2 &\xrightarrow{kk_1u^2} \varnothing \\
\mathrm{S}_2 &\xrightarrow{Fk_1u^2} \varnothing.
\end{aligned} \tag{7.25}$$

This system is widely used to study the formation of Turing patterns [107] in reaction-diffusion systems [98]. The trivial steady state of the system is $n_1 = u\Omega$ and $n_2 = 0$, where $\Omega$ is the volume of the reactor. For a fixed reactor volume, a larger $u$ hence corresponds to a larger number of

molecules in the reactor, reducing the effect of noise. In the limit of very large $u$, the kinetics of the stochastic system tends to that of the deterministic one.
The third reaction in the system is not elementary since it involves three reactant molecules. We therefore extend PSRD to also handle trimolecular reactions by using a three-dimensional partial-propensity structure and factoring out two other reactants. We choose this strategy over expanding the network into elementary reactions in order to render the parameters $k$ and $F$ comparable to the deterministic limit case. We do not use discretization-corrected propensities since no theoretical framework is available for trimolecular reactions [127].
For the simulations we fix the dimensionless constants such that $F = 0.04$ and $k = 0.06$, and we choose the macroscopic rate $k_1 = 1$. In 2D we simulate the reaction-diffusion system in a computational domain of area $\Omega = 0.64^2$, divided into $K^2 = 64^2$ subvolumes (or subareas) of edge length $h = 0.01$. At the boundary of the computational domain, periodic boundary conditions are used for the jump reactions. The resulting reaction-diffusion system in 2D thus is:

$$\left.\begin{array}{ccc}
\emptyset & \xrightarrow{Fk_1u^3} & \mathrm{S}_1^{(l,m)} \\
\mathrm{S}_1^{(l,m)} & \xrightarrow{Fk_1u^2} & \emptyset \\
\mathrm{S}_1^{(l,m)} + 2\mathrm{S}_2^{(l,m)} & \xrightarrow{k_1} & 3\mathrm{S}_2^{(l,m)} \\
\mathrm{S}_2^{(l,m)} & \xrightarrow{kk_1u^2} & \emptyset \\
\mathrm{S}_2^{(l,m)} & \xrightarrow{Fk_1u^2} & \emptyset \\
\mathrm{S}_1^{(l,m)} & \xrightarrow{D_1/h^2} & \mathrm{S}_1^{(l',m')} \\
\mathrm{S}_2^{(l,m)} & \xrightarrow{D_2/h^2} & \mathrm{S}_2^{(l',m')}
\end{array}\right\} \quad \forall\ (l,m) \in I_{\mathrm{all}}\,, \tag{7.26}$$

where $I_{\mathrm{all}}$ is the set of all possible subvolume indices in 2D and $(l', m')$ are the neighboring subvolumes of $(l, m)$ taking into account the periodic boundary conditions, hence $l' \in \{[((l-1) \pm 1 + 2K) \mod K] + 1\}$ and $m' \in \{[((m-1) \pm 1 + 2K) \mod K] + 1\}$. At $t = 0$, the initial population is:

$$\begin{aligned}
n_1^{(l,m)} &= \begin{cases} \frac{uh^2}{2} + \lfloor 0.04(r-0.5)uh^2 + 0.5 \rfloor, & \text{for } 24 \le l, m \le 40 \\ uh^2, & \text{otherwise.} \end{cases} \\
n_2^{(l,m)} &= \begin{cases} \frac{uh^2}{4} + \lfloor 0.02(r-0.5)uh^2 + 0.5 \rfloor, & \text{for } 24 \le l, m \le 40 \\ 0, & \text{otherwise,} \end{cases}
\end{aligned} \tag{7.27}$$

where $r$ is a uniform random number in $[0, 1)$ that acts as an initial perturbation. We use the diffusion constants $D_1 = 2 \cdot 10^{-5}$ and $D_2 = D_1/2$.
Figure 7.6 shows the 2D simulation results from PSRD and from a deterministic simulation. It shows the spatial concentration distribution of species $\mathrm{S}_1$, normalized with $u$, at time $t_{\mathrm{f}} = 2000/(k_1u^2)$. Figures 7.6A and 7.6B show the normalized concentration distributions for $u = 10^6$ and $10^7$, respectively, as obtained using PSRD. The maximum number of molecules of $\mathrm{S}_1$ in any subvolume is on the order of $h^2u = 0.01u$. For $u = 10^6$, approximately $0.3 \cdot 10^9$ reaction events are simulated until $t_{\mathrm{f}}$ with $f_{\mathrm{r}} \approx 0.12$ and a total runtime (CPU time) of 157 s for PSRD and 200 s for NSM. For $u = 10^7$, the number of reaction events happening during the simulation increases to $\approx 3 \cdot 10^9$ with $f_{\mathrm{r}} \approx 0.14$ and a total runtime (CPU time) of 1854 s for PSRD and 2290 s for NSM.
Increasing $u$ increases the total number of molecules in the reactor and hence decreases the noise in the system. The normalized concentration distribution obtained from a deterministic simulation is independent of $u$ and is shown in Fig. 7.6C. The deterministic simulation is done using the same

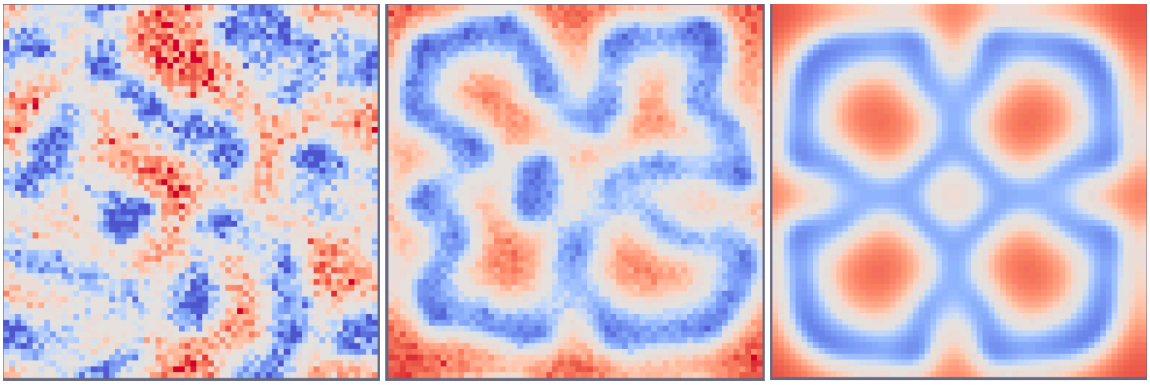



Figure 7.6: Normalized spatial concentration distribution of species $S_1$ in the two-dimensional Gray-Scott reaction-diffusion system (Eq. 7.26) for $F = 0.04$, $k = 0.06$, $k_1 = 1$, and $D_1 = 2D_2 = 2 \cdot 10^{-5}$ in a square computational domain of area $0.64^2$, divided into $N_v = 64^2$ subvolumes (or subareas) of edge length $h = 0.01$. The concentration in each subvolume is shown as a color ranging from blue (concentration zero) to red (concentration one). (A)+(B) Concentration distributions, normalized by $u$, obtained using PSRD for $u = 10^6$ and $u = 10^7$, respectively. (C) Normalized concentration distribution obtained from a deterministic simulation using the same parameters, simulated using second-order finite differences. All snapshots are taken at time $t = 2000/(k_1 u^2)$.

numerical scheme as Pearson [98] in order to render the results comparable. This is a second-order finite-difference discretization of the Laplacian for the diffusion part and a first-order explicit Euler scheme with time-step size $\Delta t = 1.125$ for time stepping. The results show that as the number of molecules in the reactor increases with increasing $u$, the stochastic spatial pattern tends toward the deterministic one. The intrinsic noise in the stochastic system, however, breaks the symmetry of the pattern.

We also simulate the reaction-diffusion system in Eq. 7.26 in 3D (using triplet indices for the subvolumes) in a computational domain of volume $\Omega = 0.64^3$, divided into $K^3 = 64^3$ subvolumes of edge length $h = 0.01$. Again using periodic boundary conditions for the diffusion, the resulting reaction-diffusion system is:

$$\left.\begin{array}{rcl}
\emptyset & \xrightarrow{Fk_1u^3} & \mathrm{S}_1^{(l,m,n)} \\
\mathrm{S}_1^{(l,m,n)} & \xrightarrow{Fk_1u^2} & \emptyset \\
\mathrm{S}_1^{(l,m,n)} + 2\mathrm{S}_2^{(l,m,n)} & \xrightarrow{k_1} & 3\mathrm{S}_2^{(l,m,n)} \\
\mathrm{S}_2^{(l,m,n)} & \xrightarrow{kk_1u^2} & \emptyset \\
\mathrm{S}_2^{(l,m,n)} & \xrightarrow{Fk_1u^2} & \emptyset \\
\mathrm{S}_1^{(l,m,n)} & \xrightarrow{D_1/h^2} & \mathrm{S}_1^{(l',m',n')} \\
\mathrm{S}_2^{(l,m,n)} & \xrightarrow{D_2/h^2} & \mathrm{S}_2^{(l',m',n')}
\end{array}\right\} \quad \forall\ (l,m,n) \in I_{\mathrm{all}}\,, \tag{7.28}$$

where $I_{\mathrm{all}}$ is the set of all possible subvolume indices in 3D and $(l', m', n')$ are the neighboring subvolumes of $(l, m, n)$ taking into account the periodic boundary conditions, hence $l' \in \{[((l-1)\pm$

$1+2K) \mod K]+1\}$, $m' \in \{[((m-1)\pm 1+2K) \mod K]+1\}$, and $n' \in \{[((n-1)\pm 1+2K) \mod K]+1\}$. At $t=0$, the initial population is:

$$\begin{aligned} n_1^{(l,m,n)} &= \begin{cases} \frac{uh^3}{2} + \lfloor 0.04(r-0.5)uh^3 + 0.5 \rfloor, & \text{for } 24 \le l,m,n \le 40 \\ uh^3, & \text{otherwise.} \end{cases} \\ n_2^{(l,m,n)} &= \begin{cases} \frac{uh^3}{4} + \lfloor 0.02(r-0.5)uh^3 + 0.5 \rfloor, & \text{for } 24 \le l,m,n \le 40 \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{7.29}$$

where $r$ is a uniform random number in $[0,1)$ that acts as an initial perturbation. We use the same diffusion constants as in the 2D case above.

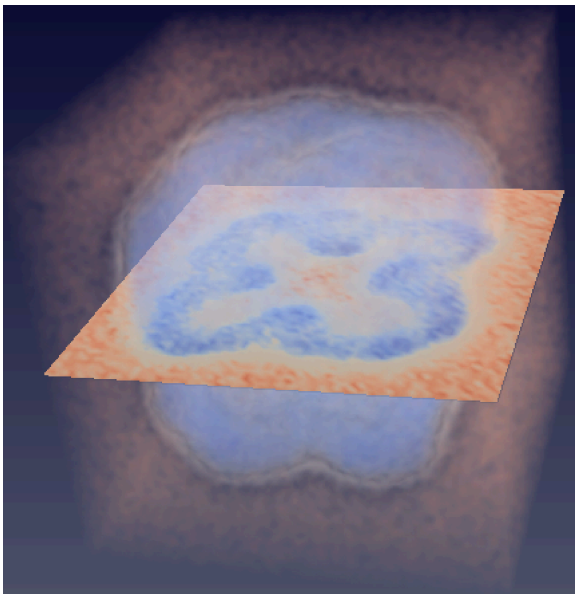


Figure 7.7: Normalized spatial concentration distribution of species $S_1$ in the three-dimensional Gray-Scott reaction-diffusion system (Eq. 7.28) for $F=0.04$, $k=0.06$, $k_1=1$, and $D_1=2D_2=2\cdot 10^{-5}$ in a cubic computational domain of volume $0.64^3$, divided into $N_v=64^3$ subvolumes of edge length $h=0.01$. The concentration in each subvolume, normalized by $u=10^8$, is shown as a color ranging from blue (concentration zero) to red (concentration one). The snapshot is taken at time $t=2000/(k_1u^2)$.

Figure 7.7 shows the 3D concentration distribution of species $S_1$ at time $t_f=2000/(k_1u^2)$, normalized with $u=10^8$. For these parameters, the maximum number of molecules of species $S_1$ in any subvolume is $uh^3=100$ and hence the intrinsic noise breaks the symmetry of the Turing pattern. Approximately $36\cdot 10^9$ reaction events are simulated until $t_f$ with $f_r \approx 0.1$. The total runtime (CPU time) for PSRD is 77 413 s, for NSM it is 100 636 s (extrapolated).

## 7.3 Conclusions and Summary

We have introduced the on-lattice partial-propensity stochastic reaction-diffusion (PSRD) method. PSRD proceeds by dividing the computational domain into $N_v$ subvolumes. The chemical reaction

system in each subvolume is assumed to be well mixed and it is imposed that molecules can only react with partners within the same subvolume. Diffusion is modeled by jump "reactions" between neighboring subvolumes. PSRD combines composition-rejection sampling [30, 18] (see Sections 2.3.7 and 2.4.5) with the concept of partial propensities (see Sec. 2.4.2). Computational efficiency is achieved by binning the subvolumes and using partial propensities to group the reactions within each subvolume.

PSRD samples trajectories from the exact solution of the reaction-diffusion master equation for on-lattice reaction-diffusion systems, provided the subvolume sizes are within admissible bounds [133, 127, 134]. This is done by first sampling the subvolume using composition-rejection sampling, and then sampling the index of the next reaction within that subvolume using linear search over the dynamically grouped partial propensities, analogous to the sorting partial-propensity direct method (SPDM) (see Sec. 2.4.4). The computational cost of PSRD to sample the next subvolume is $O(G_\mathrm{a})$, where the number of bins is $G_\mathrm{a} = \log_2(a_\mathrm{max}/a_\mathrm{min}) + 1$, $a_\mathrm{max}$ is the maximum total propensity in any subvolume, and $a_\mathrm{min}$ is the smallest non-zero total propensity in any subvolume. In any simulation, the number $G_\mathrm{a}$ scales at most as $O(\log_2 N_\mathrm{v})$. If the logarithmic span of the propensities can be *a priori* bounded by a constant, the cost of sampling the subvolume reduces to $O(1)$[18]. The computational cost to sample the index of the next reaction within a subvolume is $O(N)$, where $N$ is the number of species in the reaction network. Thus, the overall computational cost of PSRD is $O(G_\mathrm{a} + N)$, which is bounded in the worst case by $O(\log_2 N_\mathrm{v} + N)$. This cost of PSRD is independent of whether the SRD simulation is dominated by "real" reactions or by "diffusion reactions". We demonstrated this scaling of the computational cost using prototypical benchmark cases for both types of reaction networks: strongly coupled and weakly coupled.

PSRD inherits the limitations of partial-propensity methods. It is hence limited to reaction networks comprising only elementary reactions. For spatiotemporal reaction-diffusion simulations, however, including non-elementary reactions is of questionable value since no kinetic-theoretical framework exists for them [127, 35]. It is hence unclear how the propensity functions of non-elementary reactions should be correctly formulated in a discretized space [127].

Due to the more complex data structures used in partial-propensity methods, we do not expect PSRD to offer significant speed-ups for small ($N \lessapprox 10$) chemical reaction networks. In these cases, the next subvolume method [100, 119] can be as efficient or faster than PSRD. In addition, PSRD is restricted to chemical reaction networks that do not involve time delays. This could be overcome by using dPDM (delay PDM) (see Sec. 6.2.1) instead of SPDM inside each subvolume in PSRD. Our current software implementation of PSRD is moreover limited to rectangular computational domains. This limitation, however, is not inherent to the method as such and future developments will consider extending the method to computational domains of arbitrary shape [141, 142], e.g., using unstructured meshes [143].

While we have described the basic version of PSRD for simplicity and conciseness of the presentation, the algorithm can be further improved in efficiency using standard techniques. Using a binary tree search instead of linear search over subvolume bins [1], the computational cost of sampling the next subvolume can, for example, be reduced to $O(\log_2 G_\mathrm{a})$, rendering the overall computational cost of such a variant of PSRD $O(\log_2 G_\mathrm{a} + N)$ and in the worst-case $O(\log_2 \log_2 N_\mathrm{v} + N)$. Moreover, for weakly coupled reaction networks the computational cost of sampling the next reaction within a subvolume can be reduced to $O(G_\mathrm{r})$ using the partial-propensity method with composition-rejection sampling (PSSA-CR) within each subvolume. $G_\mathrm{r}$ is the logarithmic span of non-zero propensities within the subvolume. In summary, the computational cost of PSRD can be reduced to $O(\log_2 G_\mathrm{a} + N)$ or even $O(\log_2 G_\mathrm{a} + \log_2 G_\mathrm{r})$ for certain classes of reaction networks and

when using a binary search tree also within PSSA-CR. These improvements can be realized at the expense of larger memory requirements, which is why we did not include them in the presentation here. Their implementation, however, is straightforward and they will be included in future versions of the PSRD software package.

PSRD uses dynamic bubble sort for the reactions within each subvolume. This is inspired by the sorting direct method (SDM) [16] and its partial-propensity variant SPDM. Sorting SSAs have been shown to be particularly efficient on multi-scale (stiff) reaction networks where the propensities of different reactions are orders of magnitude apart. This means that a small fraction of reactions can potentially account for the majority of reaction events. The dynamic "bubbling up" of these reactions in the reaction list reduces the average search depth when sampling the next reaction as it accumulates the most frequent reactions at the top of the list. Using a sorting SSA inside each subvolume of an on-lattice SRD simulation is particularly advantageous since the propensities of different reaction types scale differently with subvolume size (see Eq. 7.2). While the propensities of bimolecular reactions scale as $\Omega_c^{-1}$, those of source reactions scale as $\Omega_c$, and the propensities of unimolecular reactions are independent of $\Omega_c$. The propensities of "diffusion reactions" scale as $h^{-2}$. Reducing the grid spacing $h$ thus renders the reaction network increasingly multi-scale with the propensity ratio between the fastest and slowest reactions scaling at most as $h^6$ in 3D subvolumes ($h^4$ in 2D subvolumes).

Taken together, PSRD offers an improved scaling of the computational cost for exact on-lattice SRD simulations. This can lead to significant performance improvements when simulating strongly coupled spatiotemporal processes, such as colloidal aggregation and scale-free biochemical networks [45, 46, 144, 40].

# Bibliography


[1] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *J. Phys. Chem. A*, vol. 104, pp. 1876–1889, 2000.

[2] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *J. Comput. Phys.*, vol. 22, pp. 403–434, 1976.

[3] D. T. Gillespie, "A rigorous derivation of the chemical master equation," *Physica A*, vol. 188, no. 1-3, pp. 404–425, 1992.

[4] S. Chapman, "On the brownian displacements and thermal diffusion of grains suspended in a non-uniform fluid," *Proc. R. Soc. Lond. A*, vol. 119, no. 781, pp. 34–53, 1928.

[5] A. Kolmogorov, "Über die analytischen methoden in der wahrscheinlichkeitsrechnung," *Math. Ann.*, vol. 104, p. 415, 1931.

[6] W. Feller, "On the integro-differential equations of purely discontinuous Markoff processes," *Trans. Amer. Math. Soc.*, vol. 48, no. 1-3, pp. 488–515, 1940.

[7] W. Feller, "On boundaries and lateral conditions for the kolmogorov differential equations," *Ann. Math.*, vol. 65, no. 3, pp. 527–570, 1957.

[8] D. A. McQuarrie, "Stochastic approach to chemical kinetics," *J. Appl. Prob.*, vol. 4, no. 3, pp. 413–478, 1967.

[9] J. L. Doob, "Topics in the theory of Markoff chains," *Trans. Amer. Math. Soc.*, vol. 52, no. 1-3, pp. 37–64, 1942.

[10] J. L. Doob, "Markoff chains – denumerable case," *Trans. Amer. Math. Soc.*, vol. 58, pp. 455–473, 1945.

[11] D. G. Kendall, "Stochastic processes and population growth," *J. R. Stat. Soc. B*, vol. 11, no. 2, pp. 230–264, 1949.

[12] M. S. Barlett, "Stochastic processes or the statistics of change," *J. R. Stat. Soc. C*, vol. 2, no. 1, pp. 44–64, 1953.

[13] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, "New algorithm for monte carlo simulation of ising spin systems," *J. Comput. Phys.*, vol. 17, no. 1, pp. 10–18, 1975.

[14] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, vol. 81, no. 25, pp. 2340–2361, 1977.

[15] Y. Cao, H. Li, and L. Petzold, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," *J. Chem. Phys.*, vol. 121, no. 9, pp. 4059–4067, 2004.

[16] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova, "The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior," *Comput. Biol. Chem.*, vol. 30, no. 1, pp. 39–49, 2006.

[17] H. Li and L. Petzold, "Logarithmic direct method for discrete stochastic simulation of chemically reacting systems," tech. rep., Department of Computer Science, University of California Santa Barbara, 2006.

[18] A. Slepoy, A. P. Thompson, and S. J. Plimpton, "A constant-time kinetic monte carlo algorithm for simulation of large biochemical reaction networks," *J. Chem. Phys.*, vol. 128, p. 205101, 2008.

[19] D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *J. Chem. Phys.*, vol. 115, no. 4, pp. 1716–1733, 2001.

[20] Y. Cao, D. T. Gillespie, and L. R. Petzold, "Avoiding negative populations in explicit Poisson tau-leaping," *J. Chem. Phys.*, vol. 123, p. 054104, 2005.

[21] Y. Cao, D. T. Gillespie, and L. R. Petzold, "Efficient step size selection for the tau-leaping simulation method," *J. Chem. Phys.*, vol. 124, p. 044109, 2006.

[22] X. Peng, W. Zhou, and Y. Wang, "Efficient binomial leap method for simulating chemical kinetics," *J. Chem. Phys.*, vol. 126, p. 224109, 2007.

[23] A. Auger, P. Chatelain, and P. Koumoutsakos, "$R$-leaping: Accelerating the stochastic simulation algorithm by reaction leaps," *J. Chem. Phys.*, vol. 125, p. 084103, 2006.

[24] X. Peng and Y. Wang, "$l$-leap: accelerating the stochastic simulation of chemically reacting systems," *Appl. Math. Mech.*, vol. 28, no. 10, pp. 1361–1371, 2007.

[25] X. Cai and Z. Xu, "$k$-leap method for accelerating stochastic simulation of coupled chemical reactions," *J. Chem. Phys.*, vol. 126, p. 074102, 2007.

[26] Y. Cao, D. T. Gillespie, and L. R. Petzold, "The slow-scale stochastic simulation algorithm," *J. Chem. Phys.*, vol. 122, p. 014116, 2005.

[27] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie, "Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method," *J. Chem. Phys.*, vol. 119, no. 24, pp. 12784–12794, 2003.

[28] R. Ramaswamy, N. González-Segredo, and I. F. Sbalzarini, "A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks," *J. Chem. Phys.*, vol. 130, no. 24, p. 244104, 2009.

[29] R. Ramaswamy and I. F. Sbalzarini, "A partial-propensity variant of the composition-rejection stochastic simulation algorithm for chemical reaction networks," *J. Chem. Phys.*, vol. 132, no. 4, p. 044102, 2010.

[30] L. Devroye, *Non-uniform random variate generation.* Springer-Verlag New York, 1986.

[31] T. Wilhelm, "Chemical systems consisting only of elementary steps - a paradigma for nonlinear behavior," *J. Math. Chem.*, vol. 27, no. 1–2, pp. 71–88, 2000.

[32] K. R. Schneider and T. Wilhelm, "Model reduction by extended quasi-steady-state approximation," *J. Math. Biol.*, vol. 40, no. 5, pp. 443–450, 2000.

[33] R. W. Hockney and J. W. Eastwood, *Computer Simulation using Particles.* Institute of Physics Publishing, 1988.

[34] L. Verlet, "Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules," *Phys. Rev.*, vol. 159, no. 1, pp. 98–103, 1967.

[35] M. von Smoluchowski, "Versuch einer mathematischen theorie der koagulationskinetik kolloider lösungen," *Z. Phys. Chem*, vol. 92, no. 2, pp. 129–168, 1917.

[36] P. G. J. van Dongen and M. H. Ernst, "Fluctuations in coagulating systems," *J. Stat. Phys.*, vol. 49, no. 5/6, pp. 879–926, 1987.

[37] P. G. J. van Dongen, "Fluctuations in coagulating systems. II," *J. Stat. Phys.*, vol. 49, no. 5/6, pp. 927–975, 1987.

[38] S. D. T. Axford, "Theoretical calculations on smoluchowski kinetics: Perikinetic reactions in highly aggregated systems," *Proc. R. Soc. Lond. A*, vol. 452, no. 1953, pp. 2355–2368, 1996.

[39] M. S. Turner, P. Sens, and N. D. Socci, "Nonequilibrium raftlike membrane domains under continuous recycling," *Phys. Rev. Lett.*, vol. 95, p. 168301, 2005.

[40] R. Albert, "Scale-free networks in cell biology," *J. Cell Sci.*, vol. 118, no. 21, pp. 4947–4957, 2005.

[41] H. Kurata, H. El-Samad, T.-M. Yi, M. Khammash, and J. Doyle, "Feedback regulation of the heat shock response in *E. coli*," in *Proc. 40th IEEE conference on Decision and Control*, pp. 837–842, 2001.

[42] H. Li, Y. Cao, L. R. Petzold, and D. T. Gillespie, "Algorithms and software for stochastic simulation of biochemical reacting systems," *Biotechnol. Prog.*, vol. 24, pp. 56–61, 2008.

[43] J. A. D. Wattis, "The modified Becker-Döring system with aggregation-dominated rate coefficients," *J. Phys. A: Math. Theor.*, vol. 42, p. 045002, 2009.

[44] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, pp. 651–654, 2000.

[45] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, pp. 268–276, 2001.

[46] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, JAN 2002.

[47] W. J. Heuett and H. Qian, "Grand canonical Markov model: a stochastic theory for open nonequilibrium biochemical networks," *J. Chem. Phys.*, vol. 124, p. 044110, 2006.

[48] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J. H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, 2003.

[49] D. T. Gillespie, "The chemical langevin equation," *J. Chem. Phys.*, vol. 113, pp. 297–306, July 2000.

[50] D. F. Anderson, A. Ganguly, and T. G. Kurtz, "Error analysis of tau-leap simulation methods," *Ann. Appl. Prob.*, 2011.

[51] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie, "Consistency and stability of tau-leaping schemes for chemical reaction systems," *Multiscale Model. Simul.*, vol. 4, no. 3, pp. 867–895, 2005.

[52] J. A. Helmuth, S. Reboux, and I. F. Sbalzarini, "Exact stochastic simulations of intra-cellular transport by mechanically coupled molecular motors," *J. Comput. Sci.*, vol. 2, no. 4, pp. 324–334, 2011.

[53] H. A. Kramers, "Brownian motion in a field of force and the diffusion model of chemical reactions," *Physica*, vol. 7, pp. 284–304, 1940.

[54] J. E. Moyal, "Stochastic processes and statistical physics," *J. R. Stat. Soc. B*, vol. 11, no. 2, pp. 150–210, 1949.

[55] S. Chandrasekhar, "Stochastic problems in physics and astronomy," *Rev. Mod. Phys.*, vol. 15, pp. 1–89, 1943.

[56] H. Haken, "Cooperative phenomena in systems far from thermal equilibrium and in nonphysical systems," *Rev. Mod. Phys.*, vol. 47, no. 1, pp. 67–121, 1975.

[57] D. Bedeaux, "Equivalence of the master equation and the Langevin equation," *Phys. Lett. A*, vol. 62, no. 1, pp. 10–12, 1977.

[58] D. T. Gillespie, "The mathematics of Brownian motion and Johnson noise," *Am. J. Phys.*, vol. 64, no. 3, pp. 225–240, 1996.

[59] D. T. Gillespie, "The multivariate langevin and fokker-planck equations," *Am. J. Phys.*, vol. 64, pp. 1246–1257, Oct. 1996.

[60] N. G. van Kampen, *Stochastic Processes in Physics and Chemistry.* North Holland, 2nd ed., 2001.

[61] W. Horsthemke and L. Brenig, "Non-linear fokker-planck equation as an asymptotic representation of the master equation," *Z. Physik*, vol. B 27, pp. 341–348, 1977.

[62] H. Qian, S. Saffarian, and E. L. Elson, "Concentration fluctuations in a mesoscopic oscillating chemical reaction system," *Proc. Natl. Acad. Sci. USA*, vol. 99, no. 16, pp. 10376–10381, 2002.

[63] H. Qian, "Open-system nonequilibrium steady state: Statistical thermodynamics, fluctuations, and chemical oscillations," *J. Phys. Chem. B*, vol. 110, no. 31, pp. 15063–15074, 2006.

[64] N. G. van Kampen, "Thermal fluctuations in nonlinear systems," *J. Math. Phys.*, vol. 4, no. 2, p. 190, 1963.

[65] H. H. McAdams and A. Arkin, "Stochastic mechanisms in gene expression," *Proc. Natl. Acad. Sci. USA*, vol. 94, no. 3, pp. 814–819, 1997.

[66] A. Arkin, J. Ross, and H. H. McAdams, "Stochastic kinetic analysis of developmental pathway bifurcation in phage $\lambda$-infected *Escherichia coli* cells," *Genetics*, vol. 149, no. 4, pp. 1633–1648, 1998.

[67] P. V. E. McClintock, "Random fluctuations - unsolved problems of noise," *Nature*, vol. 401, pp. 23–25, Sept. 1999.

[68] M. B. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335–338, 2000.

[69] N. Barkai and S. Leibler, "Biological rhythms - circadian clocks limited by noise," *Nature*, vol. 403, no. 6767, pp. 267–268, 2000.

[70] O. G. Berg, J. Paulsson, and M. Ehrenberg, "Fluctuations and quality of control in biological cells: Zero-order ultrasensitivity reinvestigated," *Biophys. J.*, vol. 79, no. 3, pp. 1228–1236, 2000.

[71] A. Eldar and M. B. Elowitz, "Functional roles for noise in genetic circuits," *Nature*, vol. 467, no. 7312, pp. 167–173, 2010.

[72] S. Engblom, "Computing the moments of high dimensional solutions of the master equation," *Appl. Math. Comput.*, vol. 180, no. 2, pp. 498–515, 2006.

[73] R. Ramaswamy, N. González-Segredo, I. F. Sbalzarini, and R. Grima, "Discreteness-induced concentration inversion in mesoscopic chemical systems," *Nat. Commun.*, vol. 3, p. 779, 2012.

[74] G. Nicolis and I. Prigogine, *Self-Organization in Nonequilibrium Systems: From Dissipative Structures to Order through Fluctuations*. John Wiley & Sons, 1977.

[75] I. Prigogine, *From Being to Becoming: Time and Complexity in the Physical Sciences*. W. H. Freeman, 1980.

[76] J. Tomita, M. Nakajima, T. Kondo, and H. Iwasaki, "No transcription-translation feedback in circadian rhythm of KaiC phosphorylation," *Science*, vol. 307, no. 5707, pp. 251–254, 2005.

[77] M. Nakajima, K. Imai, H. Ito, T. Nishiwaki, Y. Murayama, H. Iwasaki, T. Oyarna, and T. Kondo, "Reconstitution of circadian oscillation of cyanobacterial KaiC phosphorylation in vitro," *Science*, vol. 308, no. 5720, pp. 414–415, 2005.

[78] J. S. van Zon, D. K. Lubensky, P. R. H. Altena, and P. R. ten Wolde, "An allosteric model of circadian KaiC phosphorylation," *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 18, pp. 7420–7425, 2007.

[79] D. Zwicker, D. K. Lubensky, and P. R. ten Wolde, "Robust circadian clocks from coupled protein-modification and transcription-translation cycles," *Proc. Natl. Acad. Sci. USA*, vol. 107, no. 52, pp. 22540–22545, 2010.

[80] F. A. Chandra, G. Buzi, and J. C. Doyle, "Glycolytic oscillations and limits on robust efficiency," *Science*, vol. 333, no. 6039, pp. 187–192, 2011.

[81] P. H. Baxendale and P. E. Greenwood, "Sustained oscillations for density dependent Markov processes," *J. Math. Biol.*, vol. 63, no. 3, pp. 433–457, 2011.

[82] F. Schlogl, "Chemical reaction models for nonequilibrium phase-transitions," *Z. Physik*, vol. 253, no. 2, pp. 147–161, 1972.

[83] I. R. Henderson, P. Owen, and J. P. Nataro, "Molecular switches −− the ON and OFF of bacterial phase variation," *Mol. Microbiol.*, vol. 33, no. 5, pp. 919–932, 1999.

[84] E. M. Ozbudak, M. Thattai, H. N. Lim, B. I. Shraiman, and A. van Oudenaarden, "Multistability in the lactose utilization network of *Escherichia coli*," *Nature*, vol. 427, no. 6976, pp. 737–740, 2004.

[85] J. R. Pomerening, E. D. Sontag, and J. E. Ferrell, "Building a cell cycle oscillator: hysteresis and bistability in the activation of Cdc2," *Nat. Cell Biol.*, vol. 5, no. 4, pp. 346–351, 2003.

[86] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in *Escherichia coli*," *Nature*, vol. 403, no. 6767, pp. 339–342, 2000.

[87] B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Essential Cell Biology*. New York: Garland Publication, Inc., 1997.

[88] D. Bratsun, D. Volfson, L. S. Tsimring, and J. Hasty, "Delay-induced stochastic oscillations in gene regulation," *Proc. Natl. Acad. Sci. USA*, vol. 102, no. 41, pp. 14593–14598, 2005.

[89] X. Cai, "Exact stochastic simulation of coupled chemical reactions with delays," *J. Chem. Phys.*, vol. 126, p. 124108, Mar. 2007.

[90] Q. Li and X. Lang, "Internal noise-sustained circadian rhythms in a drosophila model," *Biophys. J.*, vol. 94, no. 6, pp. 1983–1994, 2008.

[91] Z. Xu and X. Cai, "Stochastic simulation of delay-induced circadian rhythms in *Drosophila*," *EURASIP J. Bioinform. Syst. Biol.*, vol. 2009, 2009.

[92] M. Barrio, K. Burrage, A. Leier, and T. Tian, "Oscillatory regulation of Hes1: Discrete stochastic delay modelling and simulation," *PLoS Comput. Biol.*, vol. 2, no. 9, pp. 1017–1030, 2006.

[93] T. Tian, K. Burrage, P. M. Burrage, and M. Carletti, "Stochastic delay differential equations for genetic regulatory networks," *J. Comput. Appl. Math.*, vol. 205, no. 2, pp. 696–707, 2007.

[94] W. Zhou, X. Peng, Z. Yan, and Y. Wang, "Accelerated stochastic simulation algorithm for coupled chemical reactions with delays," *Comput. Biol. Chem.*, vol. 32, pp. 240–242, Aug. 2008.

[95] B. Bayati, P. Chatelain, and P. Koumoutsakos, "D-leaping: Accelerating stochastic simulation algorithms for reactions with delays," *J. Comput. Phys.*, vol. 228, pp. 5908–5916, Sept. 2009.

[96] R. A. Fisher, "The wave of advance of advantageous genes," *Ann. Eugenics*, vol. 7, pp. 355–369, 1937.

[97] A. N. Kolmogorov, I. G. Petrovsky, and N. S. Piskunov, "Étude de l'équation de la diffusion avec croissance de la quantité de matière et son application à un problème biologique," *Bulletin Université d'Etat à Moscou (Bjul. Moskowskogo Gos. Univ.)*, vol. Série internationale A 1, pp. 1–26, 1937.

[98] J. E. Pearson, "Complex patterns in a simple system," *Science*, vol. 261, pp. 189–192, July 1993.

[99] M. A. J. Chaplain, M. Ganesh, and I. G. Graham, "Spatio-temporal pattern formation on spherical surfaces: numerical simulation and application to solid tumour growth," *J. Math. Biol.*, vol. 42, pp. 387–423, 2001.

[100] J. Elf and M. Ehrenberg, "Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases," *Systems Biology*, vol. 1, pp. 230–236, Dec. 2004.

[101] D. Fange and J. Elf, "Noise-induced Min phenotypes in *E. coli*," *PLoS Comput. Biol.*, vol. 2, pp. 637–648, June 2006.

[102] S. J. Altschuler, S. B. Angenent, Y. Wang, and L. F. Wu, "On the spontaneous emergence of cell polarity," *Nature*, vol. 454, no. 7206, pp. 886–889, 2008.

[103] K. Takahashi, S. Tanase-Nicola, and P. R. ten Wolde, "Spatio-temporal correlations can drastically change the response of a MAPK pathway," *Proc. Natl. Acad. Sci. USA*, vol. 107, no. 6, pp. 2473–2478, 2010.

[104] M. Bergdorf, I. F. Sbalzarini, and P. Koumoutsakos, "A Lagrangian particle method for reaction-diffusion systems on deforming surfaces," *J. Math. Biol.*, vol. 61, pp. 649–663, 2010.

[105] D. M. Holloway, F. J. P. Lopes, L. d. F. Costa, B. A. N. Travencolo, N. Golyandina, K. Usevich, and A. V. Spirov, "Gene expression noise in spatial patterning: hunchback promoter structure affects noise amplitude and distribution in drosophila segmentation," *PLoS Comput. Biol.*, vol. 7, no. 2, p. e1001069, 2011.

[106] R. D. Benguria and M. C. Depassier, "Speed of fronts of the reaction-diffusion equation," *Phys. Rev. Lett.*, vol. 77, no. 6, pp. 1171–1173, 1996.

[107] A. M. Turing, "The chemical basis of morphogenesis," *Phil. Trans. R. Soc. London*, vol. B237, pp. 37–72, 1952.

[108] A. Gierer and H. Meinhardt, "A theory of biological pattern formation," *Kybernetik*, vol. 12, pp. 30–39, 1972.

[109] A. Koch and H. Meinhardt, "Biological pattern formation: from basic mechanisms to complex structures," *Rev. Mod. Phys.*, vol. 66, pp. 1481–1507, 1994.

[110] H. Meinhardt and A. J. d. B. Piet, "Pattern formation in *Escherichia coli* : A model for the pole-to-pole oscillations of min proteins and the localization of the division site," *Proc. Natl. Acad. Sci. USA*, vol. 98, no. 25, pp. 14202–14207, 2001.

[111] C. X. Zhou, H. Y. Guo, and Q. Ouyang, "Experimental study of the dimensionality of black-eye patterns," *Phys. Rev. E*, vol. 65, p. 036118, Mar. 2002.

[112] L. Yang, M. Dolnik, A. M. Zhabotinsky, and I. R. Epstein, "Spatial resonances and superposition patterns in a reaction-diffusion model with interacting turing modes," *Phys. Rev. Lett.*, vol. 88, no. 20, p. 208303, 2002.

[113] G. H. Gunaratne, Q. Ouyang, and H. L. Swinney, "Pattern formation in the presence of symmetries," *Phys. Rev. E*, vol. 50, pp. 2802—2820, 1994.

[114] D. Panja, "Effects of fluctuations on propagating fronts," *Physics Reports*, vol. 393, pp. 87–174, 2004.

[115] C. W. Gardiner, K. J. McNeil, D. F. Walls, and I. S. Matheson, "Correlations in stochastic theories of chemical-reactions," *J. Stat. Phys.*, vol. 14, no. 4, pp. 307–331, 1976.

[116] M. Springer and J. Paulsson, "Biological physics - harmonies from noise," *Nature*, vol. 439, no. 7072, pp. 27–28, 2006.

[117] J. S. van Zon and P. R. ten Wolde, "Green's-function reaction dynamics: A particle-based approach for simulating biochemical networks in time and space," *J. Chem. Phys.*, vol. 123, p. 234910, 2005.

[118] S. S. Andrews and D. Bray, "Stochastic simulation of chemical reactions with spatial resolution and single molecule detail," *Phys. Biol.*, vol. 1, no. 3, pp. 137–151, 2004.

[119] J. Hattne, D. Fange, and J. Elf, "Stochastic reaction-diffusion simulation with MesoRD," *Bioinformatics*, vol. 21, no. 12, pp. 2923–2924, 2005.

[120] D. Rossinelli, B. Bayati, and P. Koumoutsakos, "Accelerated stochastic and hybrid methods for spatial simulations of reaction-diffusion systems," *Chem. Phys. Lett.*, vol. 451, pp. 136–140, 2008.

[121] K. A. Iyengar, L. A. Harris, and P. Clancy, "Accurate implementation of leaping in space: The spatial partitioned-leaping algorithm," *J. Chem. Phys.*, vol. 132, no. 9, p. 094101, 2010.

[122] L. Ferm, A. Hellander, and P. Lötstedt, "An adaptive algorithm for simulation of stochastic reaction-diffusion processes," *J. Comput. Phys.*, vol. 229, pp. 343–360, 2010.

[123] W. Koh and K. T. Blackwell, "An accelerated algorithm for discrete stochastic simulation of reaction-diffusion systems using gradient-based diffusion and tau-leaping," *J. Chem. Phys.*, vol. 134, no. 15, p. 154103, 2011.

[124] M. Jeschke, R. Ewald, and A. M. Uhrmacher, "Exploring the performance of spatial stochastic simulation algorithms," *J. Comp. Phys.*, vol. 230, no. 7, pp. 2562–2574, 2011.

[125] M. J. Morelli and P. R. ten Wolde, "Reaction brownian dynamics and the effect of spatial fluctuations on the gain of a push-pull network," *J. Chem. Phys.*, vol. 129, no. 5, p. 054112, 2008.

[126] S. Hellander and P. Lotstedt, "Flexible single molecule simulation of reaction-diffusion processes," *J. Comput. Phys.*, vol. 230, pp. 3948–3965, May 2011.

[127] D. Fange, O. G. Berg, P. Sjoberg, and J. Elf, "Stochastic reaction-diffusion kinetics in the microscopic limit," *Proc. Natl. Acad. Sci. USA*, vol. 107, no. 46, pp. 19820–19825, 2010.

[128] P. Gray and S. K. Scott, "Autocatalytic reactions in the isothermal, continuous stirred tank reactor - isolas and other forms multistability," *Chem. Eng. Sci.*, vol. 38, no. 1, pp. 29–43, 1983.

[129] P. Gray and S. K. Scott, "Autocatalytic reactions in the isothermal, continuous stirred tank reactor : Oscillations and instabilities in the system A + 2B $\rightarrow$ 3B; B $\rightarrow$ C," *Chem. Eng. Sci.*, vol. 39, no. 6, pp. 1087–1097, 1984.

[130] P. Gray and S. K. Scott, "Sustained oscillations and other exotic patterns of behavior in isothermal reactions," *J. Phys. Chem.*, vol. 89, no. 1, pp. 22–32, 1985.

[131] K. J. Lee, W. D. McCormick, Q. Ouyang, and H. L. Swinney, "Pattern formation by interacting chemical fronts," *Science*, vol. 261, no. 5118, pp. 192–194, 1993.

[132] R. Erban and S. J. Chapman, "Reactive boundary conditions for stochastic simulations of reaction-diffusion processes," *Phys. Biol.*, vol. 4, no. 1, pp. 16–28, 2007.

[133] R. Erban and S. J. Chapman, "Stochastic modelling of reaction-diffusion processes: algorithms for bimolecular reactions," *Phys. Biol.*, vol. 6, no. 4, p. 046001, 2009.

[134] Y. Kuramoto and T. Yamada, "Pattern formation in oscillatory chemical reactions," *Progr. Theoret. Phys.*, vol. 56, pp. 724–740, 1976.

[135] R. Ramaswamy, I. F. Sbalzarini, and N. González-Segredo, "Noise-induced modulation of the relaxation kinetics around a non-equilibrium steady state of non-linear chemical reaction networks," *PLoS ONE*, vol. 6, no. 1, p. e16045, 2011.

[136] R. Ramaswamy and I. F. Sbalzarini, "Intrinsic noise alters the frequency spectrum of mesoscopic oscillatory chemical reaction systems," *Sci. Rep.*, vol. 1, p. 154, 2011.

[137] B. L. Fox, "Generating markov-chain transitions quickly: I," *ORSA J. Comput.*, vol. 2, no. 2, pp. 126–135, 1990.

[138] S. Rajasekaran and K. W. Ross, "Fast algorithms for generating discrete random variates with changing distributions," *ACM Trans. Model. Comput. Simul.*, vol. 3, no. 1, pp. 1–19, 1993.

[139] T. Hagerup, K. Mehlhorn, and I. Munro, "Optimal algorithms for generating time-varying discrete random variates," in *Lect. Notes Comput. Sci.*, vol. 700, pp. 253–264, Springer, New York, 1993.

[140] J. Hattne, "The algorithms and implementation of MesoRD," master thesis, Department of Information Technology, Uppsala University, 2006.

[141] I. F. Sbalzarini, A. Mezzacasa, A. Helenius, and P. Koumoutsakos, "Effects of organelle shape on fluorescence recovery after photobleaching," *Biophys. J.*, vol. 89, no. 3, pp. 1482–1492, 2005.

[142] I. F. Sbalzarini, A. Hayer, A. Helenius, and P. Koumoutsakos, "Simulations of (an)isotropic diffusion on curved biological surfaces," *Biophys. J.*, vol. 90, no. 3, pp. 878–885, 2006.

[143] S. Engblom, L. Ferm, A. Hellander, and P. Lötstedt, "Simulation of stochastic reaction-diffusion processes on unstructured meshes," *SIAM J. Sci. Comput.*, vol. 31, no. 3, pp. 1774–1797, 2009.

[144] A. L. Barabási and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nat. Rev. Genet.*, vol. 5, no. 2, pp. 101–113, 2004.