POLITECNICO DI TORINO

Master's Degree
in Mathematical Engineering

Master's Degree Thesis

Challenges in simulation of Chemical Reaction Networks

Supervisor
prof. Enrico Bibbona

Candidate
Gabriele Galilei

Academic Year 2023-2024

Summary

The study of deterministic models of Chemical Reaction Networks has been a central topic in Chemical Physics since the 1970s thanks to the work of M. Feinberg, F. Horn, and R. Jackson. The stochastic counterpart was formulated in the same period and investigated mostly through simulations. Stochastic simulation algorithms have been developed, the most known one being the exact simulation method usually associated to the name of Gillespie who made it popular in the field of Chemical Physics. A mathematical theory of stochastic models of Chemical Reaction Networks has been developed more recently, but efficient simulation methods are still needed to address most models that remains intractable with analytical tools.

The interest in studying complicated models and the availability of an increased computer power made it possible to simulate larger examples and it stimulated the development of approximate simulation methods to speed up the computations further. The most known approximated simulation algorithm is the so-called τ-leaping method, again due to Gillespie. Recently, several authors proposed improvements to τ-leaping algorithm, including the MidPoint corrections and Post-Leap Checks both introduced by D. Anderson and collaborators on which we focus.

In this thesis, we develop a few non-trivial numerical examples where the application of such improved τ-leaping methods are compared among each other and against exact simulations to evaluate their performance.

The first setting is that of the stochastic Lotka-Volterra predator-prey model, in a parameter regime where the extinction of one of the populations is a rare event, but still possible when the initial population consistency is not extremely large. This is an interesting classical case where we expect that the trivial τ-leaping method can easily fail, while the improved ones should show their capabilities. We illustrate our findings showing that both improvements (MidPoint corrections and Post-Leap Checks) are important and that they can be successfully combined.

In the second setting we simulate a stochastic model of nanoparticle growth that has been subject of recent studies. In this case the simulation algorithm needs a careful implementation due to several source of complexity, both in terms of speed and memory usage, and in terms of the peculiarity of the model. In this case MidPoint corrections are not applicable, while Post-Leap Checks are strictly necessary for the τ-leaping method to be applicable. However, in some interesting parameter regimes, the computational advantage provided by the approximated τ-leaping method with respect to the exact Gillespie algorithm partly fades away due to the high rate of failure of the Post-Leap checks.

Contents

Introduction

In 1864, C. M. Guldberg and P. Waage developed a theory, known as the *affinity theory*, in which the rate of a chemical reaction is directly proportional to the concentration of its reactants [1]. Only in 1879 did they microscopically justify this assumption by attributing it to the frequency of particle collisions [2]. This gave rise to the law of mass action. The significance of this discovery led to the development of various kinetic theories on the concentration of chemical species based on the law of mass action. In 1900, J. H. van't Hoff's studies on the thermodynamic implications of the law of mass action and on solutes, C. N. Hinshelwood and N. N. Semenov's research on critical reactions and the concept of a *chain reaction*, and R. Aris' work leading to a rigorous mathematical formulation of reaction systems and the introduction of the concept of Chemical Reaction Network followed. Subsequently, in the early 1970s, F. J. M. Horn and R. Jackson [3] and M. Feinberg [4] developed the theory now known as Chemical Reaction Network theory.

With the advent of computers, there naturally arose a desire to simulate the behavior of chemical component concentrations in a Chemical Reaction Network. In [5], D. Gillespie introduced the Stochastic Simulation Algorithm, better known as the Gillespie algorithm, for the exact simulation of Chemical Reaction Networks. Subsequently, many steps were taken to approximate such algorithms. These methods are still used to describe the behavior of biochemical systems and more.

In this work, the aim was to compare various simulation methods for Chemical Reaction Networks in term of computational time and strong approximation error; then possible modifications to those algorithms were studied. All of the latter is then applied on two case studies of different mathematical and applied nature.

Chapter 1 introduces the basics of Chemical Reaction Network theory and its notation. Simultaneously, the law of mass action and kinetics based on it are explained, which will be used in the case studies. Previous literature results are presented, such as the possible rewriting of the trigger number of a reaction in terms of variable-rate Poisson processes [6] and the Markovian nature of stochastic processes based on Chemical Reaction Networks [7]. It is also noted that in chemistry, often the discussion revolves around the concentration of a species in the examined environment rather than particle counting. Therefore, classical scaling is introduced, modifying reaction rates to maintain the theory of Chemical Reaction Networks but acting on concentrations rather than counts.

A theorem is stated and proven by T. G. Kurtz and S. N. Either [8], imposing that for infinite-volume environments, simulations obtained through the examined methods should tend toward the deterministic solution of the associated system of ODEs for the Chemical Reaction Network.

Chapter 2 explains the exact simulation method for Chemical Reaction Networks developed by D. Gillespie [5], based on the result that concentration evolution is well described by multivariate Poisson processes.

Chapter 3 explains the approximate simulation method proposed by Gillespie himself in [9], called the τ-leaping method. The idea behind this method is to consider a time interval larger than the time between one reaction and the next one and to estimate how many times each reaction occurred in that interval. However, this model often led to negative concentrations that make mathematical sense but not logical sense. Therefore, in [9] [10] [11], an adaptive time interval is introduced, respecting the so-called *leap condition*, thus reducing the probability of failure. Subsequently, a better implementation of the τ-leaping algorithm is explained, where each proposed leap respects the leap condition, hence its name Post-Leap Check. Finally, the MidPoint correction is described, an additional approximation term in the evaluation of the reaction rates to use in τ-leaping methods.

In Chapter 4, the algorithms described in Chapters 2 and 3 are applied to the Lotka-Volterra predator-prey model [12] [13], chosen because a deterministic solution of the associated ODEs can be described. The associated Chemical Reaction Network is derived, simulated, and the strong approximation errors and average computational times are analyzed.

In Chapter 5, the analysis performed in Chapter 4 is repeated, but applied to the case study of the growth and nucleation of gold nanoparticles. In particular, the number of nanoparticles for each size is simulated until the system reaches a steady state, i.e., until the exhaustion of gold monomers. The analysis is also repeated in a scaling different from the classical one, allowing the nucleation and growth probabilities to be of the same order from the early instants.

Part I

Preliminary concepts

Chapter 1

Chemical Reaction Networks

1.1 Introduction to Chemical Reaction Networks

Definition 1.1.1 (Chemical Reaction Networks)**.** We call *Chemical Reaction Network* the triplet $\{\mathscr{S}, \mathscr{C}, \mathscr{R}\}$ where:

(i) $\mathscr{S} = \{S_1, \ldots, S_L\}$ is the set of the *species*;

(ii) $\mathscr{C}$ is the set of the *complexes*, non-negative linear combinations of species with integer coefficients;

(iii) $\mathscr{R} = \{y_k \longrightarrow y'_k : y_k, y'_k \in \mathscr{C} \; e \; y_k \neq y'_k\}$ is the set of the possible reactions.

Thinking in a biochemical context, the species are nothing but the different types of molecules involved in the reactions, and the complexes are combinations of these chemical species with their respective stoichiometric coefficients appearing in a reaction on the left or right side.
Note that we will denote by:

(i) L the number of species;

(ii) M the number of reactions.

Example 1.1.1. Taking into account the simple reaction:

$$\begin{aligned} A + B &\longrightarrow 2A \\ B &\longrightarrow A \end{aligned}$$

we will have that:

- $\mathscr{S} = \{A,\ B\}$;
- $\mathscr{C} = \{A,\ B,\ A+B,\ 2A\}$;
- $\mathscr{R} = \{A+B \longrightarrow 2A,\ B \longrightarrow A\}$.

Given a fixed reaction k, this can be written as follows:

$$\sum_{i=1}^{L} y_{ki} S_i \longrightarrow \sum_{i=1}^{L} y'_{ki} S_i$$

where the arrays $y_k, y'_k \in \mathbb{Z}^L \geq 0$ represent the coefficient vectors associated with the reactant complex and the product complex, respectively.

Definition 1.1.2 (Reaction vector)**.** The *reaction vector* ζ_k associated with the k-th reaction is defined as $\zeta_k := y'_k - y_k \in \mathbb{Z}^L$.

Example 1.1.2. Referring to the previous example 1.1.1, the reaction vectors will be:

$$\zeta_1 = (2-1,\, 0-1) = (1,\, -1) \quad \zeta_2 = (1-0,\, 0-1) = (1,\, -1).$$

Specifically, species in the left complex are referred to as *reactants*, whereas species in the right one are termed *products*.

Definition 1.1.3 (Reaction Rate)**.** Given the reaction $y \to y'$, the term *reaction rate*, *reaction speed*, or *propensity function* $\lambda_{y \to y'} : \mathbb{N}_0^L \longrightarrow [0, \infty)$ is used to describe the speed at which the reaction $y \to y'$ occurs. The vector of reaction rates is referred to as the *kinetics*.

The evolution in time of most of the biochemical systems can be expressed via a Chemical Reaction Network, paired with a kinetics.

The most common example of kinetics is that of *mass action*.

Example 1.1.3 (Mass Action Kinetics)**.** Given a reaction of the form:

$$\sum_{i=1}^{L} y_i S_i \longrightarrow \sum_{i=1}^{L} y'_i S_i,$$

and adopting the convention that $0^0 = 1$, the *mass action kinetics* is defined as follows:

$$\lambda_{y \to y'}(x) = \kappa \cdot \prod_{i=1}^{L} \frac{x_i!}{(x_i - y_i)!}$$

where κ is referred to as the *reaction rate constant*. Thus, for the reaction $2A + B \longrightarrow C$ we will have that $\lambda([a, b, c]) = \kappa \cdot a(a-1) \cdot b$.

In particular, when the concentration of a chemical species, denoted as x_i, is less than y_i, the reaction rate $\lambda_{y \to y'}(x)$ is zero. This circumstance indicates a deficiency of reactants and subsequently the impossibility of the chemical reaction. Conversely, as the concentrations of reactants increase, the likelihood of the reaction demonstrates an enhanced yield, reflecting in a higher reaction rate. This kinetic phenomenon adheres to the law of mass action, and the resulting kinetics are consequently categorized as *mass action-type kinetics*.

1.2 Results on stochastic processes based on Chemical Reaction Networks

The stochastic process $X = (X_1, \ldots, X_L)$, representing the number of particles in the system over time for each species, can be formally characterized using the reaction vectors $\zeta_1, \ldots, \zeta_M$ associated with the possible reactions.

Let N_k be the counting process related to how many times the k-th reaction has been triggered in the time interval $(0, t)$. Hence it is trivial to think that:

$$X(t) = X(0) + \sum_{k=1}^{M} N_k(t)\, \zeta_k.$$

Consider Y_k a unit-rate Poisson Process and λ_k the propensity function of the k-th reaction. In [6], Kurtz formally describes how, under certain conditions, the solution to the following problem exists and it is unique:

$$N_k(t) = Y_k \left(\int_0^t \lambda_k(X(s))\, ds \right). \tag{1.1}$$

This result expresses how any counting process can be written as a unit-rate Poisson process with variable time. In particular, if the propensity function of the k-th reaction in the state $\lambda_k(X(s))$ is high for $s \in (0, t)$, then the time argument of the Poisson distribution is also high since these functions are non-negative. This leads to higher counts. Conversely, if $\lambda_k(X(s))$ is low for $s \in (0, t)$, so will be the counts.

Therefore the dynamics of the process can be described as follows:

$$X(t) = X(0) + \sum_{k=1}^{M} N_k(t)\, \zeta_k \tag{1.2}$$

$$= X(0) + \sum_{k=1}^{M} Y_k \left(\int_0^t \lambda_k(X(s))\, ds \right) \zeta_k. \tag{1.3}$$

Simultaneously, this also makes the process X a continuous-time Markov chain, presenting an interesting aspect from a simulation perspective: when simulating the process X in the time window $(t, t+r)$, it is not necessary to store information about the process's past, only its current state, $X(t)$.

1.3 Classical Scaling

In the chemical context, the description of a system state often revolves around chemical concentrations rather than the number of particles, primarily due to the large number of particles involved, which is on the order of Avogadro's number ($N_A \approx 6{\cdot}10^{23}$). Specifically, the definition $V := N_A \cdot v$ is introduced, where v represents the volume of the container in which reactions take place. So *Classical Scaling* is when there is:

(i) an initial abundance of species on the order of $V \gg 1$;

(ii) a modification of rate constants to account for the dimensions of the surrounding environment: each rate constant of reaction k from y to y' is approximately of the order $c_k^V = \mathscr{O}(V^{1-s_k})$, where s_k denotes the number of reacting particles involved in reaction k.

Consequently, the reaction rates λ_k^V and the state X^V, which now reflects not the number of particles but rather the concentration of species expressed in moles per unit volume, are updated. Thus, we have:

$$X^V(t) = X^V(0) + \frac{1}{V}\sum_{k=1}^{M} Y_k \left(\int_0^t \lambda_k^V(X^V(s))\, ds \right) \zeta_k \tag{1.4}$$

This scaling procedure is derived from simplifying assumptions: considering a reaction of the type $A + B \to C$, the probability of collision between two particles of species A and B will vary proportionally to the volume, and this holds generally for higher-order reactions. For reactions of the type $A \to C$, on the other hand, the probability of occurrence in containers of different sizes does not undergo significant variations and, therefore, does not exhibit a volume dependence.

One can define a system of coupled ordinary differential equations based on a Chemical Reaction Network:

$$\dot{x} = F(x), \quad F(x) := \sum_{k=1}^{M} \lambda_k(x)\zeta_k. \tag{1.5}$$

This system, yielding continuous solutions, pertains to the concentrations of the species. An illustrative example follows.

Example 1.3.1. Let's revisit example 1.1.1. The reaction rates are $\lambda_1([a,b]) = \kappa_1 \cdot a \cdot b$ and $\lambda_2([a,b]) = \kappa_2 \cdot b$. The reaction vectors are $\zeta_1 = (1,-1)$ and $\zeta_2 = (1,-1)$. Thus, the equations take the form:

$$\begin{pmatrix} \dot{a} \\ \dot{b} \end{pmatrix} = \kappa_1 ab \begin{pmatrix} 1 \\ -1 \end{pmatrix} + \kappa_2 b \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\begin{cases} \dot{a} = \kappa_1 \cdot a \cdot b + \kappa_2 \cdot b \\ \dot{b} = -\kappa_1 \cdot a \cdot b - \kappa_2 \cdot b \end{cases}$$

It is worth noting that the reverse operation from a system of ordinary differential equations to a Chemical Reaction Network is not unique, even though the simulation effects coincide.

Regarding the relation between the solution of 1.5 and the stochastic process X^V, in [8], the following theorem is stated and proven. Given the tedious notation, we will consider:

$$|v| = |v_1| + |v_2| + \dots.$$

and

$$\tilde{Y}(u) := Y(u) - u$$

the Poisson process Y centered with respect to its mean, called *centered* or *compounded Poisson process*.

Theorem 1.3.1 (Etemadi's inequality)**.** Let $X_1, \ldots, X_n$ be independent random variables and let $S_k := X_1 + \ldots + X_k$ the partial sum. Then for every $\alpha \geq 0$ it is true that:

$$\mathbb{P}\left[\max_{0\leq k\leq n} |S_k| \geq 3\alpha\right] \leq \max_{0\leq k\leq n} \mathbb{P}\left[|S_k| \geq \alpha\right].$$

The proof is in [**?**].

Lemma 1.3.1 (Poisson law of large numbers)**.** Let $\tilde{Y}$ be a unit-rate Poisson process. It is true for all $v \geq 0$ that:

$$\lim_{V\to\infty} \sup_{u\leq v} \left|\frac{1}{V}\tilde{Y}(Vu)\right| = 0, \quad a.s..$$

Remark 1.3.1. The lemma just stated is a uniform version of the law of large numbers for centered Poisson processes; indeed, neglecting the upper limit, one would have:

$$\begin{aligned}
\lim_{V\to\infty} \frac{1}{V}|Y(Vu)| &\leq \lim_{V\to\infty} \frac{1}{V}\left(|Y(u) - 0| + |Y(2u) - Y(u)| + \ldots + |Y(Vu) - Y((V-1)u)|\right) \\
&= \lim_{V\to\infty} \frac{1}{V} \sum_{i=0}^{V-1} |Y((i+1)u) - (i+1)u - Y(iu) + iu| \\
&= \lim_{V\to\infty} \frac{1}{V} \sum_{i=0}^{V-1} |Y((i+1)u) - Y(iu) - u| = 0
\end{aligned}$$

which turns out to be a regular law of large numbers for centered Poisson processes.

Proof. Let us consider the stochastic process $\frac{1}{V}\tilde{Y}(Vu)$. Firstly, let us prove that $\tilde{Y}(t)$ is a continuous-time martingale.

(i) $Y(t)$ is adapted to $\{Y(s)\}_{s\leq t}$ by assumption, and thus the process $\tilde{Y}(t)$ will also be adapted as it is a deterministic translation of an adapted process.

(ii) $\mathbb{E}[|\tilde{Y}(t)|] \leq \mathbb{E}[|Y(t)| + |t|] = 2t < \infty$.

(iii)

$$\begin{aligned}
\mathbb{E}[\tilde{Y}(t) - \tilde{Y}(s)|\tilde{Y}(s)] &= \mathbb{E}[Y(t) - t - Y(s) + s|Y(s)] \\
&= \mathbb{E}[Y(t) - Y(s)|Y(s)] - (t+s) \\
&= \mathbb{E}[Y(t) - Y(s)] - (t+s) \\
&= (t-s) - (t-s) = 0.
\end{aligned}$$

Let us define:

$$\Xi_j^h := \bigcup_{i=0}^{2^j h} \left\{ \frac{i}{2^j} \right\}$$

so that $\Xi_0^h = \{0, 1, \ldots, h\}$, $\Xi_1^h = \{0, 1/2, 1, \ldots, h\}$, $\Xi_2^h = \{0, 1/4, 1/2, 3/4, 1, \ldots, h\}$. So $\lim_{j\to\infty} \Xi_j^h$ is dense in $[0, h]$. Since $\tilde{Y}$ has almost surely right-continuous trajectories, then for all $v > 0$, for all $V \in \mathbb{N}$:

$$\sup_{u<v} |\tilde{Y}(Vu)| = \lim_{j\to\infty} \max_{u\in\Xi_j^v} |\tilde{Y}(Vu)| \ \ a.s..$$

It is true that $\Xi_j^h \subset \Xi_{j+1}^h$ for all j. Therefore, by continuity of the probability measure:

$$\begin{aligned}
\mathbb{P}\left[\sup_{u<v} |\tilde{Y}(Vu)| > V\varepsilon\right] &= \mathbb{P}\left[\lim_{j\to\infty} \max_{u\in\Xi_j^v} |\tilde{Y}(Vu)| > V\varepsilon\right] \\
&= \lim_{j\to\infty} \mathbb{P}\left[\max_{u\in\Xi_j^v} |\tilde{Y}(Vu)| > V\varepsilon\right].
\end{aligned}$$

Since $\tilde{Y}$ is a translated jump process, it can be considered as a partial sum of random variables. Hence Etemadi's inequality 1.3.1 can be applied:

$$\mathbb{P}\left[\max_{u\in\Xi_j^v} |\tilde{Y}(Vu)| > V\varepsilon\right] \leq 3 \max_{u\in\Xi_j^v} \mathbb{P}\left[|\tilde{Y}(Vu)| > \frac{V\varepsilon}{3}\right].$$

and so:

$$\mathbb{P}\left[\sup_{u<v} |\tilde{Y}(Vu)| > V\varepsilon\right] \leq 3 \lim_{j\to\infty} \max_{u\in\Xi_j^v} \mathbb{P}\left[|\tilde{Y}(Vu)| > \frac{V\varepsilon}{3}\right].$$

For all $0 < u < v$ and for all $\beta \in (0,1)$:

$$\begin{aligned}
\mathbb{P}\left[|\tilde{Y}(Vu)| > \frac{V\varepsilon}{3}\right] &\leq \mathbb{P}\left[\tilde{Y}(Vu) > \frac{V\varepsilon}{3}\right] + \mathbb{P}\left[-\tilde{Y}(Vu) > \frac{V\varepsilon}{3}\right] \\
&= \mathbb{P}\left[\tilde{Y}(Vu) > \frac{V\varepsilon}{3}\right] + \mathbb{P}\left[-\tilde{Y}(Vu) > \frac{V\varepsilon}{3}\right] \\
&= \mathbb{P}\left[e^{V^{\beta-1}\tilde{Y}(Vu)} > e^{\frac{V^\beta\varepsilon}{3}}\right] + \mathbb{P}\left[e^{-V^{\beta-1}\tilde{Y}(Vu)} > e^{\frac{V^\beta\varepsilon}{3}}\right].
\end{aligned}$$

Markov's inequality states that if X is a non-negative process then $\mathbb{P}[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$. Applying it we have that:

$$\begin{aligned}
\mathbb{P}\left[|\tilde{Y}(Vu)| > \frac{V\varepsilon}{3}\right] &\leq e^{-\frac{V^\beta\varepsilon}{3}} \left(\mathbb{E}\left[e^{V^{\beta-1}\tilde{Y}(Vu)}\right] + \mathbb{E}\left[e^{-V^{\beta-1}\tilde{Y}(Vu)}\right]\right) \\
&= e^{-\frac{V^\beta\varepsilon}{3}} \left(\mathbb{E}\left[e^{V^{\beta-1}Y(Vu)}\right] e^{-v^\beta u} + \mathbb{E}\left[e^{-V^{\beta-1}Y(Vu)}\right] e^{V^\beta u}\right).
\end{aligned}$$

The moment generating function of the Poisson distribution is $\mathbb{E}[e^{tY}] = e^{\lambda(e^t-1)}$, from which:

$$\begin{aligned}\mathbb{P}\left[|\tilde{Y}(Vu)| > \frac{V\varepsilon}{3}\right] &\leq e^{-\frac{V^\beta\varepsilon}{3}}\left(e^{Vu\left(e^{V^{\beta-1}}-1\right)}e^{-V^\beta u} + e^{Vu\left(e^{-V^{\beta-1}}-1\right)}e^{V^\beta u}\right)\\ &= e^{-\frac{V^\beta\varepsilon}{3}}\left(e^{Vu\left(e^{V^{\beta-1}}-1-V^{\beta-1}\right)} + e^{Vu\left(e^{-V^{\beta-1}}-1+V^{\beta-1}\right)}\right)\\ &\leq 2e^{-\frac{V^\beta\varepsilon}{3}}e^{Vu\left(e^{V^{\beta-1}}-1-V^{\beta-1}\right)}.\end{aligned}$$

Since as $V \to \infty$ then $V^{\beta-1} \to 0$ it is possible using Taylor's expansion of $e^{V^{\beta-1}}$ to write:

$$e^{V^{\beta-1}} - V^{\beta-1} - 1 = \frac{V^{2\beta-2}}{2}e^{V^{\beta-1}} + o(V^{2\beta-2})$$

and so:

$$\begin{aligned}\mathbb{P}\left[|\tilde{Y}(Vu)| > \frac{V\varepsilon}{3}\right] &\leq 2e^{-\frac{V^\beta\varepsilon}{3}}e^{Vu\frac{V^{2\beta-2}}{2}e^{V^{\beta-1}}}\\ &\leq 2e^{-\frac{V^\beta\varepsilon}{3}}e^{Vv\frac{V^{2\beta-2}}{2}e^{V^{\beta-1}}}\\ &\leq 2e^{-\frac{V^\beta\varepsilon}{3}}e^{Vv\frac{V^{2\beta-2}}{2}e}.\end{aligned}$$

Choosing $\beta = \frac{1}{2}$:

$$\mathbb{P}\left[|\tilde{Y}(Vu)| > \frac{V\varepsilon}{3}\right] \leq 2e^{-\frac{\sqrt{V}\varepsilon}{3}}e^{\frac{ve}{2}}.$$

So:

$$\begin{aligned}\mathbb{P}\left[\sup_{u<v}|\tilde{Y}(Vu)| > V\varepsilon\right] &\leq 6\lim_{j\to\infty}\max_{u\in\Xi_j^v}e^{-\frac{\sqrt{V}\varepsilon}{3}}e^{\frac{ve}{2}}\\ &= 6e^{-\frac{\sqrt{V}\varepsilon}{3}+\frac{ve}{2}}.\end{aligned}$$

Therefore as $V \to \infty$ then $\mathbb{P}\left[\sup_{u<v}|\tilde{Y}(Vu)| > V\varepsilon\right] \to 0$, hence the thesis. □

Theorem 1.3.2. Suppose $K \subset E \subset \mathbb{R}_+^L$ compact such that:

$$\sum_{k=1}^{M}|\zeta_k|\sup_{x\in K}\lambda_k^V(x) < \infty \tag{1.6}$$

and that it exists $M_K > 0$ such that:

$$|F(x) - F(y)| \leq M_K|x-y|, \quad x, y \in K.$$

Let x be a solution to the problem 1.5, and let X^V be a stochastic process satisfying 1.4 such that $\lim_{V\to\infty}X^V(0) = x(0)$. Then for all $T \geq 0$ it is true that:

$$\lim_{V\to\infty}\sup_{t\leq T}\left|X^V(t) - x(t)\right| = 0 \quad a.s.. \tag{1.7}$$

Proof. It is worth noting that the assumption of the existence of a global solution for the problem 1.5 is justified by the hypothesis of Lipschitz continuity of the function F in the theorem statement and by the continuity of the reaction rates. These, in addition to ensuring existence, also guarantee the global uniqueness of the solution x.

Given $T \geq 0$, $\lambda_k^V(x(T))$ depends only on the values of x in a small temporal neighborhood preceding T, given the Markovian nature of the process. Therefore, the validity of $\sum_{k=1}^{M} |\zeta_k| \sup_{x \in K} \lambda_k^V(x) < \infty$ depends only on the values $\{x(t) : t \leq T\}$. Let us fix a point $x \in E$, then there exists a compact $K \subset E$ such that $x \in E$. Hence it is true that 1.6. Therefore this holds for all $x \in E$ and it is permissible to take the weak condition:

$$\sum_{k=1}^{M} |\zeta_k| \, \bar{\lambda}_k^V < \infty$$

where $\bar{\lambda}_k^V := \sup_{x \in E} \lambda_k^V(x)$.

Since for all $K \subset E$ it exists $M_K > 0$ such that it holds $|F(x) - F(y)| \leq M_K |x - y|$ for every $x, y \in K$, then it exists $M \geq \max_{K \subset E} M_K$ such that it is true that:

$$|F(x) - F(y)| \leq M |x - y|, \quad x, y \in E.$$

Denoting $\tilde{Y}_k(u) := Y_k(u) - u$ as the centered Poisson process Y_k with respect to its mean, X^V can be expressed in the form:

$$\begin{aligned} X^V(t) &= X^V(0) + \frac{1}{V} \sum_{k=1}^{M} \left(\tilde{Y}_k \left(\int_0^t \lambda_k^V(X^V(s))\, ds \right) + \int_0^t \lambda_k^V(X^V(s))\, ds \right) \zeta_k \\ &= X^V(0) + \frac{1}{V} \sum_{k=1}^{M} \tilde{Y}_k \left(\int_0^t \lambda_k^V(X^V(s))\, ds \right) + \int_0^t \left(\sum_{k=1}^{M} \zeta_k \frac{\lambda_k^V(X^V(s))}{V} \right) ds \\ &= X^V(0) + \frac{1}{V} \sum_{k=1}^{M} \tilde{Y}_k \left(\int_0^t \lambda_k^V(X^V(s))\, ds \right) + \int_0^t F(X^V(s))\, ds. \end{aligned}$$

It is trivial how $F\left(\int_0^t X(s)\,ds\right) = \int_0^t F(X(s))\,ds$ for $X = X^V$ or $X = x$. So:

$$\begin{aligned}
\left|X^V(t) - x(t)\right| &= \Big|X^V(t) - \left(x(0) + \int_0^t F(x(s))\,ds\right) + \\
&\quad + X^V(0) - X^V(0) + F\left(\int_0^t X^V(s)\,ds\right) - F\left(\int_0^t X^V(s)\,ds\right)\Big| \\
&= \Big|X^V(0) - x(0) + X^V(t) - X^V(0) - \int_0^t F(X^V(s))\,ds + \\
&\quad + F\left(\int_0^t X^V(s)\,ds\right) - \int_0^t F(x(s))\,ds\Big| \\
&\leq \left|X^V(0) - x(0)\right| + \left|X^V(t) - X^V(0) - \int_0^t F(X^V(s))\,ds\right| + \\
&\quad + \left|F\left(\int_0^t X^V(s)\,ds\right) - F\left(\int_0^t x(s)\,ds\right)\right| \\
&\leq \left|X^V(0) - x(0)\right| + \left|X^V(t) - X^V(0) - \int_0^t F(X^V(s))\,ds\right| + \\
&\quad + M\left|\int_0^t X^V(s)\,ds - \int_0^t x(s)\,ds\right| \\
&\leq \left|X^V(0) - x(0)\right| + \left|X^V(t) - X^V(0) - \int_0^t F(X^V(s))\,ds\right| + \\
&\quad + M\int_0^t \left|X^V(s) - x(s)\right|\,ds.
\end{aligned}$$

Let us define:

$$\varepsilon^V(T) := \sup_{t \leq T} \left|X^V(t) - X^V(0) - \int_0^t F(X^V(s))\,ds\right|$$

Therefore:

$$\left|X^V(t) - x(t)\right| \leq \left|X^V(0) - x(0)\right| + \varepsilon^V(t) + M\int_0^t \left|X^V(s) - x(s)\right|\,ds.$$

and applying Gronwall's Lemma we have that:

$$\left|X^V(t) - x(t)\right| \leq \left(\left|X^V(0) - x(0)\right| + \varepsilon^V(t)\right) e^{Mt}. \tag{1.8}$$

Coming back to the definition of ε^V:

$$\begin{aligned}
\varepsilon^V(T) &:= \sup_{t\leq T}\left|X^V(t) - X^V(0) - \int_0^t F(X^V(s))\,ds\right| \\
&= \sup_{t\leq T}\left|\frac{1}{V}\sum_{k=1}^{M}\tilde{Y}_k\left(\int_0^t \lambda_k^V(X^V(s))\,ds\right)\zeta_k\right| \\
&\leq \sup_{t\leq T}\frac{1}{V}\sum_{k=1}^{M}\left|\tilde{Y}_k\left(\int_0^t \lambda_k^V(X^V(s))\,ds\right)\zeta_k\right| \\
&\leq \sup_{t\leq T}\frac{1}{V}\sum_{k=1}^{M}\left|\tilde{Y}_k\left(\int_0^t \lambda_k^V(X^V(s))\,ds\right)\right||\zeta_k| \\
&\leq \sup_{t\leq T}\frac{1}{V}\sum_{k=1}^{M}\left(\left|Y_k\left(\int_0^t \lambda_k^V(X^V(s))\,ds\right)\right| + \left|-\int_0^t \lambda_k^V(X^V(s))\,ds\right|\right)|\zeta_k| \\
&= \sup_{t\leq T}\frac{1}{V}\sum_{k=1}^{M}\left(Y_k\left(\int_0^t \lambda_k^V(X^V(s))\,ds\right) + \int_0^t \lambda_k^V(X^V(s))\,ds\right)|\zeta_k| \\
&\leq \frac{1}{V}\sum_{k=1}^{M}\left(Y_k\left(\int_0^T \bar{\lambda}_k^V\,ds\right) + \int_0^T \bar{\lambda}_k^V\,ds\right)|\zeta_k| \\
&= \frac{1}{V}\sum_{k=1}^{M}\left(Y_k\left(\bar{\lambda}_k^V T\right) + \bar{\lambda}_k^V T\right)|\zeta_k|
\end{aligned}$$

Let us choose $u = \frac{1}{V}\int_0^t \lambda_k(X^V(s))\,ds$ and $v = \frac{1}{V}\int_0^T \lambda_k(X^V(s))$ and let us notice that for $t \leq T$ it is true that $u \leq v$. Then applying Lemma 1.3.1 we have:

$$\begin{aligned}
\lim_{V\to\infty}\varepsilon^V(T) &= \lim_{V\to\infty}\sup_{t\leq T}\frac{1}{V}\sum_{k=1}^{M}\left|\tilde{Y}_k\left(\int_0^t \lambda_k^V(X^V(s))\,ds\right)\right||\zeta_k| \\
&= \sum_{k=1}^{M}\left(\lim_{V\to\infty}\sup_{t\leq T}\frac{1}{V}\left|\tilde{Y}_k\left(\int_0^t \lambda_k^V(X^V(s))\,ds\right)\right|\right)|\zeta_k| \\
&= 0.
\end{aligned}$$

Taking the superior limit over $t \leq T$ and considering the limit for $V \to \infty$ for both terms of 1.8, we have:

$$\lim_{V\to\infty}\sup_{t\leq T}\left|X^V(t) - x(t)\right| \leq \lim_{V\to\infty}\sup_{t\leq T}\left(\left|X^V(0) - x(0)\right| + \varepsilon^V(t)\right)e^{Mt} = 0$$

hence the thesis. □

This allows simulating with approximate methods stochastic processes based on Chemical Reaction Networks and confirming their validity.

Part II

Simulation methods for Chemical Reaction Network

Chapter 2

Exact methods: Gillespie algorithm

Exact simulation methods allow the direct simulation of Markov chains without approximations. One of the most commonly used exact methods in the simulation of Chemical Reaction Networks is the Gillespie method.

Let X be the continuous-time Markov chain subject to simulation, and let $\{T_j\}_j$ be its transition times. We define the *embedded* discrete-time Markov chain $Z_j = X(T_j)$, which, along with the waiting times $\{W_j\}_j := \{T_{j+1} - T_j\}_j$ (exponentially distributed), provides a complete description of the dynamics of the chain X.

The *Gillespie algorithm* or *Stochastic Simulation Algorithm* (SSA), popularized by D. Gillespie [5], is designed to simulate the new state of the chain by determining which reaction will be triggered first and the corresponding waiting time before that reaction takes place.

Considering a transition time T_j, it is evident that each reaction occurs with a probability directly proportional to the corresponding propensity function in the state $x = X(T_j)$. Normalizing this probability with respect to the sum of all reaction rates, the probability that the triggered reaction at time T_j is the l-th one is given by:

$$\frac{\lambda_l(x)}{\sum_{k=1}^{M} \lambda_k(x)}$$

where $\lambda_l(x)$ denotes the propensity function for the l-th reaction at the state x such that $X(T_j) = x$.

What is left to point out is the stochastic computation of the waiting time for the next reaction to be triggered: this waiting time is the expected value of the minimum among all the waiting times for every reaction to occur. It is relevant to note that the waiting times are exponentially distributed with parameter $\lambda_k(x)$ where $x = X(T_j)$. So, denoting

by $\tilde{W}_1, \tilde{W}_2, \ldots, \tilde{W}_M$ the waiting times associated for every reaction to occur and by W_{j+1} the minimum of this times, assuming that those waiting times are mutually independent, we have:

$$\begin{aligned}\mathbb{P}[W_{j+1} \geq t] &= \prod_{k=1}^{M} \mathbb{P}[\tilde{W}_k \geq t] \\ &= \prod_{k=1}^{M} e^{-\lambda_k(x)\cdot t} \\ &= e^{-\left(\sum_{k=1}^{M} \lambda_k(x)\right)\cdot t}\end{aligned}$$

from which it emerges that $W_{j+1} \sim \textit{Exponential}\left(\sum_{k=1}^{M} \lambda_k(x)\right)$.
A realization of this random variable, denoted as τ, is then generated, yielding the subsequent jump time $t_{j+1} = t_j + \tau$.

This is the basic idea behind the Gillespie algorithm:

Algorithm 2.1 Gillespie algorithm

Consider a Chemical Reaction Network with propensity functions λ_k and reaction vectors ζ_k for $k = 1, \ldots, M$. Initialize $j = 0$, $t_0 = 0$ and $X(t_0) = x_0$.
Repeat the following steps until $t_j \geq t_{\max}$:

1: For all $k = 1, \ldots, M$ compute the reaction rates $\lambda_k(x_j)$;
2: Compute $\bar{\lambda}(x_j) := \sum_{k=1}^{M} \lambda_k(x_j)$;
3: Compute the next jump time:
   a: Generate r_1, the realization of a random variable following an exponential distribution with parameter $\bar{\lambda}(x_j)$;
   b: Update the time: $t_{j+1} \leftarrow t_j + r_1$;
4: Compute which reaction will be triggered:
   a: Generate r_2, the realization of a random variable following a uniform distribution between 0 and $\bar{\lambda}(x_j)$;
   b: Find $\mu \in \{1, \ldots, M\}$ such that:

$$\sum_{k=1}^{\mu-1} \lambda_k(x_j) \leq r_2 \leq \sum_{k=1}^{\mu} \lambda_k(x_j);$$

   the reaction μ is the triggered one;
5: Update the state: $x_{j+1} = x_j + \zeta_\mu$;
6: $j \leftarrow j + 1$.

Chapter 3

Approximated methods: τ-leaping method

In highly complex systems, such as many biochemical systems, exact methods for simulating Chemical Reaction Networks become computationally expensive. This is because the number of realizations of generated random variables scales linearly with the number of reactions that have occurred. Conditioned on $X(t) = x$, the expected waiting time for the next reaction is $\Delta_t = 1/\sum_k \lambda_k(x)$; for a high abundance of particles, reaction rates become very high, and thus $\Delta_t \ll 1$, resulting in prohibitively increased costs.

In [9], Gillespie introduced an approximate and accelerated variant of the Stochastic Simulation Algorithm (SSA) called the *τ-leaping method*. Reaction times recorded using the Gillespie algorithm have both advantages and disadvantages: while they provide a high level of detail and precision, they come with a significant computational cost and often result in numerous events being stored unnecessarily. The τ-leaping approach aims to address this issue by partitioning the temporal axis of the process history into contiguous intervals. Freezing the reaction rates as their at the initial time of an interval and calculating the number of times reactions occur in the latter allows proposing an interval-reaction pair for updating the state, a concept known as a *leap*.
The method is named after the time interval, namely τ.

It has been previously shown that it is possible to write:

$$X(t+\tau) = X(t) + \sum_{k=1}^{M} Y_k \left(\int_t^{t+\tau} \lambda_k(X(s))\, ds \right) \zeta_k.$$

Assuming that in the time window $(t, t+\tau)$ a reduced number of reactions occur, it seems logic to consider the propensity functions as approximately constant during the

time interval. So:

$$\begin{aligned} X(t+\tau) &= X(t) + \sum_{k=1}^{M} Y_k \left(\int_t^{t+\tau} \lambda_k(X(s))\, ds \right) \zeta_k \\ &= X(t) + \sum_{k=1}^{M} Y_k \left([(t+\tau) - t] \cdot \lambda_k(X(t)) \right) \zeta_k \\ &= X(t) + \sum_{k=1}^{M} Y_k \left(\tau \cdot \lambda_k(X(t)) \right) \zeta_k. \end{aligned}$$

The latter assumption is crucial for the use of the τ-leaping method and is called the *leap condition*. This assumption is what relates the classical τ-leaping method to the Euler method for solving integrals or differential equations, which is why it will be later referred to as *Euler τ-leaping*.

The simplest version of the algorithm chooses τ as a constant. The algorithm follows:

Algorithm 3.1 τ-leaping algorithm

Consider a Chemical Reaction Network with propensity functions λ_k and reaction vectors ζ_k for $k = 1, \ldots, m$. Choose a fixed value for τ and initialize $j = 0$, $t_0 = 0$ and $X(t_0) = x_0$.

Repeat the following steps until $t_j \geq t_{\max}$:

1: For all $k = 1, \ldots, M$ compute the reaction rates $\lambda_k(x_j)$;
2: Generate n_k, the realization of a random variable following a Poisson distribution with parameter $\lambda_k(x_j) \cdot \tau$;
3: Update the time: $t_{j+1} = t_j + \tau$;
4: Update the state: $x_{j+1} = x_j + \sum_{k=1}^{M} n_k \cdot \zeta_\mu$;
5: $j \leftarrow j + 1$.

Clearly, the closer τ gets to an actual reaction time, the more the approximate method approaches the exact one, leading to a more accurate but at the same time slower simulation. Additionally, since reaction rates are computed at time t_j but used over the interval $(t_j, t_j + \tau)$, it is possible to compute more reactions than the theoretically possible ones, resulting in negative particle counts. Indeed, choosing a larger τ would violate the *leap condition*, leading to a coarser approximation.

Possible improvements to the algorithm are outlined below.

3.1 Adaptive time interval for τ-leaping method

In [9], a mathematical form of the *leap condition* is presented. Let N_k be the random variable that accounts for the number of times the k-th reaction has been triggered in the time interval $(t_j, t_j + \tau)$, let ε such that $0 < \varepsilon \ll 1$ be a control parameter. Suppose we

are in state x_j. Define $\bar{\lambda}(x_j) = \sum_{k=1}^{M} \lambda_k(x_j)$. So:

$$|\Delta\lambda_k(x_j)| := \left|\lambda_k\left(x_j + \sum_{k=1}^{M} N_k\zeta_k\right) - \lambda_k(x_j)\right| \leq \varepsilon\bar{\lambda}(x_j), \quad \forall k = 1, \ldots, M. \tag{3.1}$$

The actual meaning is that the relative change of the reaction rates with respect to the sum of all the rates has to be very small in order to consider the approximation of the reaction rates as their evaluation at the beginning of the time interval valid.
It might be more straightforward to consider using $\lambda_k(x_j)$ in the second term, but this poses issues if the reaction rate is very close to zero due to a deficiency of reactants in the k-th reaction, since this would result in never accepting a leap.

By fixing the control parameter ε it is possible to find the maximum time interval τ that satisfies the *leap condition* [10]. However, since N_k are random variables, there is no guarantee that the condition is satisfied, leading to the need to repeat the leap.

Therefore, we define:

$$\begin{aligned} f_{kk'}(x) &:= \sum_{i=1}^{L} \frac{\partial\lambda_k(x)}{\partial x[i]}\zeta_{k'}[i] \\ \mu_k(x) &:= \sum_{k'=1}^{M} f_{kk'}(x)\lambda_{k'}(x) \\ \sigma_k^2(x) &:= \sum_{k'=1}^{M} f_{kk'}^2(x)\lambda_{k'}(x) \end{aligned}$$

and we require that the random variable $\Delta\lambda_k(x_j)$ must almost surely take values in:

$$[\tau\mu_k(x_j) - \sqrt{\tau\sigma_k^2(x_j)},\ \tau\mu_k(x_j) + \sqrt{\tau\sigma_k^2(x_j)}],$$

In order to satisfy the *leap condition*, this interval must be a subset of:

$$[-\varepsilon\bar{\lambda}(x_j), \varepsilon\bar{\lambda}(x_j)].$$

We then impose:

$$|\tau\mu_k(x_j)| \leq \varepsilon\bar{\lambda}(x_j) \quad \text{and} \quad \sqrt{\tau}\sigma_k(x_j) \leq \varepsilon\bar{\lambda}(x_j) \tag{3.2}$$

resulting in:

$$\tau = \min_{k=1,\ldots,M} \left\{ \frac{\varepsilon\bar{\lambda}(x_j)}{|\mu_k(x_j)|}, \frac{\varepsilon^2\bar{\lambda}^2(x_j)}{\sigma_k^2(x_j)} \right\}. \tag{3.3}$$

In [11], the above has been revisited to make an improvement. It is observed that reaction rates change by at least a minimal amount. For example, if it is a unimolecular reaction with $\lambda_k = \kappa_k x_i$, then this minimal amount will be κ_k, considering x_i is a natural number. Therefore, 3.1 is modified to:

$$|\Delta\lambda_k(x_j)| \leq \max\{\varepsilon\lambda_k(x_j), \kappa_k\} \tag{3.4}$$

this leading to:

$$\tau' := \min_{k=1,\dots,M} \left\{ \frac{\max\{\varepsilon\lambda_k(x_j), \kappa_k\}}{|\mu_k(x_j)|}, \frac{\max\{\varepsilon\lambda_k(x_j), \kappa_k\}^2}{\sigma_k^2(x_j)} \right\}.$$

Nonetheless, the scientific community exploited the difficulty to compute $\mu_k(x_j)$ and $\sigma_k^2(x_j)$. The *order* of a reaction is the sum of the stoichiometric coefficients of the reactants in the reaction. It's called *highest order of reaction* $\mathrm{HOR}(i)$ with respect to the species i the maximum of all the orders of reaction in which the species i appears as a reactant. Analyzing the different type of reaction and reflecting on their linear, quadratic or cubic behavior, it is possible to simplify the computation of $\mu_k(x_j)$ and $\sigma_k^2(x_j)$. This was theorized in [10].The leap condition has been reconsidered based on the variation of the number of particles: this must be at most one unit per species and at most $\varepsilon X_i(t_j)/g_i$ where g_i is defined as follows:

(i) if $\mathrm{HOR}(i) = 1$, take $g_i = 1$;

(ii) if $\mathrm{HOR}(i) = 2$ and each reaction requires at most 1 particle of the i-th species, take $g_i = 2$;

(iii) if $\mathrm{HOR}(i) = 2$ and there is a reaction that requires 2 particles of the i-th species, take $g_i = 2 + 1/(x[i] - 1)$;

(iv) if $\mathrm{HOR}(i) = 3$ and each reaction requires at most 1 particle of the i-th species, take $g_i = 3$;

(v) if $\mathrm{HOR}(i) = 3$ and there is a reaction that requires 2 particles of the i-th species, take $g_i = 3 + \frac{1}{2(x[i]-1)}$;

(vi) if $\mathrm{HOR}(i) = 3$ and there is a reaction that requires 3 particles of the i-th species, take $g_i = 3 + \frac{1}{x[i]-1} + \frac{2}{x[i]-2}$;

(vii) reactions with more than three reactants are extremely rare and so not taken into account.

Even though, for reactions with more than two reactants, these coefficients need to be computed for each *leap*, this computational effort is still lower compared to finding the maximum value that satisfies 3.1.

Starting from 3.4 and applying the insights gained so far, we obtain:

$$|\Delta X_j[i]| \leq \max\{\varepsilon x_j[i]/g_i, 1\} \tag{3.5}$$

As seen before, it is possible to write $\Delta X_j[i] = \sum_{k=1}^{M} N_k \zeta_k[i]$; so:

$$\mathbb{E}[\Delta X_j[i]] = \sum_{k=1}^{M} \mathbb{E}[N_k]\zeta_k[i] = \tau \sum_{k=1}^{M} \lambda_k(x_j)\zeta_k[i];$$
$$Var[\Delta X_j[i]] = \sum_{k=1}^{M} \mathrm{Var}[N_k]\zeta_k[i]^2 = \tau \sum_{k=1}^{M} \lambda_k(x_j)\zeta_k[i]^2.$$

Analogously to what was seen for τ', the following are defined:

$$\hat{\mu}_i(x) := \sum_{k=1}^{M} \zeta_k[i] \cdot \lambda_k(x);$$
$$\hat{\sigma}_i^2(x) := \sum_{k=1}^{M} \zeta_k[i]^2 \cdot \lambda_k(x);$$

Therefore:

$$\tau'' := \min_{i=1,\dots,L} \left\{ \frac{\max\{\varepsilon x[i]/g_i,1\}}{|\hat{\mu}_i(x)|}, \frac{\max\{\varepsilon x[i]/g_i,1\}^2}{\hat{\sigma}_i^2(x)} \right\}. \tag{3.6}$$

This will be the adaptive step-size used for the results obtained in Chapter 4 and Chapter 5.

3.2 τ-leaping method with Post-Leap Check

Despite the choice of the adaptive time interval 3.6, there are known cases in which the τ-leaping procedure described has still led to negative values of the chain [14] [15]. For this reason, Tian and Burrage [16] and Chatterjee and Vlachos [14] have developed an alternative approach: if the concentration of a species approaches zero, the process will no longer be simulated using Poisson random variables but instead using binomial random variables. In fact, it is sufficient to calculate the maximum number of times reaction k can occur in the current state and set it as the parameter for the binomial distribution, which cannot take values greater than this maximum due to its limited domain.

In [15], an alternative proposal suggests choosing an integer n_c between 2 and 20 and dividing the reactions into two distinct groups: non-critical reactions, for which more than n_c events are possible in the current state of the reactants, and critical reactions, for which the number of possible events in the current state is less than n_c. The algorithm performs a Gillespie method for critical reactions and a τ-leaping method for non-critical reactions; if a negative value is obtained, the leap is rejected and repeated.

Despite the validity of such methods from a simulation perspective, those method changes the actual distribution of the sample path and they may implement leaps that lead to negative values. This implies that the leap condition is violated, and it is not only a symptom of how this occurs near 0 but also an indication of how it happens even when the chain has highly positive values without showing any sign.

In [17], Anderson indeed proposes a simulation method for Chemical Reaction Networks based on the τ-leaping method introduced by Gillespie [10]. Most importantly, this method never violates the leap condition 3.1.

It has been proven that the internal times are mutually independent, and the increments are independent of the current state of the system [7]. Given the state $X(s) = z$,

we recall the definition of internal time $T_k(t) = \int_0^t \lambda_k(x)\,ds$ and define $C_k := Y_k(T_k(t))$ as the number of times the k-th reaction has been triggered until time t; N_k is, instead, the realization of a Poisson random variable with parameter $\tau\lambda_k(X(T_k(t)))$, representing the number of times the k-th reaction has been triggered in the time interval $(T_k, T_k + \lambda_k(X(T_k))\tau)$. The state of the system at time $t+\tau$ can thus be approximated by $X(t+\tau) \approx X(t) + \sum_{k=1}^{M} N_k\zeta_k$, while $Y_k(t + \lambda_k(X(T_k))\tau) = C_k + N_k$. At this point, it is necessary to check whether the proposed leap violates the leap condition or not: if indeed the leap is rejected, then choose $\tau^* < \tau$ and repeat the procedure with a slight variation.

Theorem 3.2.1. Let $Y(t)$ be a Poisson process with parameter λ and let $0 \le s < u < t$. Then it is true:

$$Y(u) - Y(s) \,|\, Y(t) = a, Y(s) = b \sim \textit{Binomial}\left(a - b, \frac{u-s}{t-s}\right).$$

Proof. Without loss of generality, it can be assumed that $s = 0$ and $Y(0) = 0$; for $t > u > 0$ we have that $Y(t) = N$. Then:

$$\begin{aligned}
\mathbb{P}[Y(u) - Y(s) = j | Y(t) = N, Y(s) = 0] &= \mathbb{P}[Y(u) = j | Y(t) = N] \\
&= \mathbb{P}[Y(u) = j, Y(t) = N]/\mathbb{P}[Y(t) = N] \\
&= \mathbb{P}[Y(t) = N | Y(u) = j]\mathbb{P}[Y(u) = j]/\mathbb{P}[Y(t) = N] \\
&= \mathbb{P}[Y(t) - Y(u) = N - j]\mathbb{P}[Y(u) = j]/\mathbb{P}[Y(t) = N] \\
&= \frac{e^{-\lambda(t-u)}(\lambda(t-u))^{N-j}}{(N-j)!}\frac{e^{-\lambda u}(\lambda u)^j}{j!}\frac{N!}{e^{-\lambda t(\lambda t)^N}} \\
&= \frac{N!}{j!(N-j)!}e^{-\lambda(t-u+u-t)}\lambda^{N-j+j-N}\left(\frac{t-u}{t}\right)^{N-j}\left(\frac{u}{t}\right)^j \\
&= \binom{N}{j}\left(\frac{u}{t}\right)^j\left(1-\frac{u}{t}\right)^{N-j}
\end{aligned}$$

□

Therefore, it is legitimate to write, with $\lambda_k = \lambda_k(X(T_k))$, that:

$$Y_k(T_k + \lambda_k\tau^*) - Y_k(T_k) \,|\, Y(T_k) = C_k, Y(T_k + \lambda_k\tau) = N_k + C_k \sim \textit{Binomial}\left(N_k, \frac{\tau^*}{\tau}\right).$$

Hence, it is sufficient to simulate this distribution and to check that it satisfies the leap condition. If the leap is again rejected, we save the number of initiations in $(T_k, T_k + \lambda_k\tau^*)$, choose $\tau^{**} < \tau^*$, and we simulate the binomial distribution, but conditioning on $Y_k(T_k)$ and $Y_k(T_k + \lambda_k\tau^*)$, since increments of the Poisson processes are mutually independent. We repeat this process until a leap is accepted.

It remains to clarify how to choose the time interval τ:

- if the *leap* is rejected then the time interval is reduced by multiplying it by p such that $0 < p < 1$;
- if the *leap* is accepted and:

- if the *leap* would have been accepted even using $\frac{3}{4}\varepsilon$ as control parameter then the time interval is increased by raising it to the power of q such that $0 < q < 1$;
- if the *leap* would have been rejected using $\frac{3}{4}\varepsilon$ as control parameter then the time interval is reduced by multiplying it by p^* such that $0 < p < p^* < 1$.

By doing so, if the leap condition is widely satisfied, the interval is increased, whereas if the leap condition is violated or if its satisfaction depends closely on the chosen control parameter, a safety margin is maintained, effectively reducing the interval. This selection mechanism is advantageous in terms of computational cost: indeed, it is illogical to find the maximum interval satisfying the leap condition when it depends strictly on an arbitrarily chosen control parameter.

New notation is introduced: for each Poisson process Y_k, an associated matrix S_k is defined; the first column contains the internal times of the chain in ascending order, and the second column contains the number of initiations up to the corresponding internal time from the first column. Thus, $Y_k(S_k(i,1)) = S_k(i,2)$. Additionally, the first row of S_k always contains the internal time $T_k = S_k(1,2)$ related to the last accepted leap, and $C_k = Y_k(T_k) = S_k(1,2)$.

From the above considerations, the following algorithm is derived:

@algorithm 3.2 τ-leaping algorithm with Post-Leap Check

Consider a Chemical Reaction Network with reaction rates λ_k and reaction vectors ζ_k for $k = 1, \dots, M$ with initial condition $X(t_0) = x_0$ where $t_0 = 0$. Initialize $T_k = C_k = 0$ for all $k = 1, \dots, M$; so $S_k = [0,0]$. Fix $0 < \varepsilon \ll 1$, $0 < p < p^* < 1$, $0 < q < 1$.

1: Compute τ as in 3.6;

Repeat the following steps until $t \geq t_{\max}$ or $\bar{\lambda}(x) \leq 0$:

2: Let B_k be the number of rows of S_k;

3: If $\lambda_k(X(T_k))\tau + T_k \geq S_k(B_k,1)$:

a: Generate

$$N_k \sim \mathit{Poisson}\,(T_k + \lambda_k(x)\tau - S_k(B_k,1)) + S_k(B_k,2) - C_k;$$

b: Let $\mathrm{row}_k = B_k$;

4: Otherwise:

a: Find I_k such that:

$$S_k(I_k - 1,1) \leq T_k + \lambda_k(x)\tau < S_k(I_k,1)$$

meaning the internal times already saved, among which the new leap is placed;

b: Compute:

$$r = \frac{T_k + \lambda_k(x)\tau - S_k(I_k - 1,1)}{S_k(I_k,1) - S_k(I_k - 1,1)};$$

c: Generate:

$$N_k \sim \mathit{Binomial}\,(S_k(I_k,2) - S_k(I_k - 1,2), r) + S_k(I_k - 1,2) - C_k;$$

d: Let $\text{row}_k = I_k - 1$;
5: Check if the *leap condition* 3.5 is verified;
6: If the leap verifies 3.5:
a: Update S_k: all rows of S_k with an index less than or equal to row_k are eliminated, and all rows from $\text{row}_k + 1$ to B_k are moved to the beginning of the matrix. Add a new initial row $[T_k + \lambda_k(x)\tau, C_k + N_k]$ to the matrix;
b: Update time $t \leftarrow t + \tau$;
c: Update $T_k \leftarrow T_k + \lambda_k(x)\tau$ e $C_k \leftarrow C_k + N_k$;
d: Update τ: if 3.5 is satisfied also for $3\varepsilon/4$, then $\tau \leftarrow \tau^q$; otherwise $\tau = p^*\tau$;
e: Update the state $x \leftarrow x + \sum_{k=1}^{M} N_k\zeta_k$;
7: If the leap does not satisfy 3.5:
a: Update S_k:add the row $[T_k + \lambda_k(x)\tau, C_k + N_k]$ between the rows with indices row_k and $\text{row}_k + 1$ in S_k, with the convention that if $\text{row}_k = B_k$, the row is simply added at the end of the matrix;
b: Update $\tau \leftarrow p\tau$.

3.3 MidPoint correction

In [18], it is investigated how, as $\tau \to 0$, the expected value and variance of the Euler τ-leaping method have a local truncation error with respect to an exact method of order $\mathscr{O}(\tau^2)$. Furthermore, in [19], it is described how, as $\tau \to 0$, the Euler τ-leaping method has a strong approximation error in 2-norm of order $\frac{1}{2}$. However, such studies have been conducted for $\tau \to 0$. For this values the use of approximate methods becomes inconvenient: indeed, these methods are useful only if the time interval under consideration is much larger than the time between two reactions, i.e., if:

$$\tau \gg \frac{1}{\sum_{k=1}^{M} \lambda_k(x)}. \tag{3.7}$$

given the state $x = X(t)$. If the underlying idea of the Euler τ-leaping method is to treat the reaction rates $\lambda_k(x)$ as constants in the time interval $(t, t+\tau)$, it would certainly seem more accurate to evaluate the reaction rates to be used in $(t, t+\tau)$ by approximating what the state might be at the midpoint of the interval, namely:

$$\rho(x) = x + \frac{\tau}{2} \sum_{k=1}^{M} \lambda_k(x)\zeta_k. \tag{3.8}$$

This method is the so called *MidPoint τ-leaping* method and it assumes that the underlying differential equations of the Chemical Reaction Network are more closely followed.

It is noteworthy that $\rho(x)$ is nothing but the second-order approximation of $X(t+\tau)$, while $X(t+\tau) \approx X(t)$ is the first-order approximation. In [18] and [19], this method was not considered because, for $\tau \to 0$, the MidPoint correction term $\frac{\tau}{2}\sum_{k=1}^{M} \lambda_k(X(t))\zeta_k$, of order $\mathscr{O}(\tau^2)$, is neglected and has no real numerical effect. In [20], it is shown that for values of τ satisfying 3.7, on the other hand, the MidPoint correction has a reducing

effect on the strong approximation error compared to an exact method. The analysis is conducted for fixed τ, but it is easy to imagine how this can bring about significant improvements even in the case of adaptive time interval.

Part III

Application of the methods

The computations were carried out using an Intel Core i7 (7th generation) processor with 16GB of RAM. The processor used is quad-core and supports Hyper-Threading, allowing each core to handle two threads simultaneously, resulting in a total of eight logical threads. The base frequency of the processor is approximately 2.8 GHz with the ability to reach higher speeds through Turbo Boost.

All code and algorithm implementations were developed using the Julia programming language, version 1.9.3. Julia was chosen for its high performance, crucial for achieving efficient and fast computations in the context of stochastic process simulation. Additionally, Julia's syntax and parallel computing capabilities facilitated the development of scalable and optimized code for this study. The *Random.jl* library was used for the generation of pseudo-random numbers.

Chapter 4

Application to Lotka-Volterra model

4.1 Introduction to the model

The *Lotka-Volterra model* (1925-26) [12] [13], also known as the *prey-predator model*, describes the dynamics of an ecosystem where only two animal species interact: a prey and a predator. In its simplest form, the model is described by a system of first-order nonlinear differential equations:

$$\begin{cases} \dot{x} = \alpha x - \beta xy \\ \dot{y} = \delta xy - \gamma y \end{cases}, \tag{4.1}$$

where x is the prey concentration, y is the predator concentration, α is the constant birth rate of preys, β is the constant death rate of preys due to predators, δ is the constant birth rate of predators if there is prey availability, and γ is the constant death rate of predators.

These equations generate oscillations in the concentrations of prey and predator, delayed among themselves, and whose form changes with variations in the system's rate constants.

Based on 4.1 it is possible to derive a Chemical Reaction Network:

$$\begin{aligned} X &\xrightarrow{\kappa_1} 2X \\ X + Y &\xrightarrow{\kappa_2} Y \\ X + Y &\xrightarrow{\kappa_3} 2Y \\ Y &\xrightarrow{\kappa_4} \emptyset \end{aligned} \tag{4.2}$$

Particularly, the first reaction models the reproduction of prey, the second the consumption of prey by predators, the third the reproduction of predators in the presence of preys,

and the fourth the death of predators. Using the definition in 1.5, one can construct from 4.2 a system of first-order ordinary differential equations, which will be:

$$\begin{cases} \dot{x} = \kappa_1 x - (\kappa_2 + \kappa_3)xy \\ \dot{y} = \kappa_3 xy - \kappa_4 y \end{cases}, \tag{4.3}$$

The comparison between this equations and the Lotka-Volterra model 4.1 leads to the conclusion that the latter can be described by the Chemical Reaction Network 4.2 under the assumption that $\kappa_1 = \alpha$, $\kappa_2 = \beta - \delta$, $\kappa_3 = \delta$, $\kappa_4 = \gamma$. The resulting Chemical Reaction Network is:

$$\begin{aligned} X &\xrightarrow{\alpha} 2X \\ X + Y &\xrightarrow{\beta-\delta} Y \\ X + Y &\xrightarrow{\delta} 2Y \\ Y &\xrightarrow{\gamma} \emptyset \end{aligned} \tag{4.4}$$

From here, the reaction rates can be derived:

$$\begin{aligned} &\lambda_1(x, y) = \alpha x; \\ &\lambda_2(x, y) = (\beta - \delta)xy; \\ &\lambda_3(x, y) = \delta xy; \\ &\lambda_4(x, y) = \gamma y; \end{aligned}$$

and the reaction vectors:

$$\zeta_1 = (1,0); \quad \zeta_2 = (-1,0); \quad \zeta_3 = (-1,1); \quad \zeta_4 = (0,-1).$$

As previously outlined, it is possible to determine a deterministic solution of this model through the analysis of the associated differential equations. Therefore, it has been used as a reference parameter to assess the effectiveness of the simulation methods that will be discussed in the course of the analysis. This possibility stems from a result illustrated in theorem 1.3.2. To achieve this, it is necessary to scale the simulated processes through *classical scaling*, and thus, the following are adopted:

$$\alpha^V = \alpha; \quad (\beta - \delta)^V = \frac{\beta - \delta}{V}; \quad \delta^V = \frac{\delta}{V}; \quad \gamma^V = \gamma;$$

and $X^V(0) = x(0)$ where $x(0)$ is the initial condition of the deterministic problem. Clearly, since the simulations are based on the number of particles, to obtain the concentrations plot, and thus to X^V, it is necessary to divide the ordinates by V.

The model used for the following simulations is determined by the choice of constants: $\alpha = 2.0$, $\beta = 0.5$, $\delta = 0.2$, $\gamma = 2.0$.

The choice of the Lotka-Volterra model given the aforementioned parameter selection is not arbitrary: the deterministic solution of such a model indeed exhibits significant

oscillations in the concentrations of prey and predators, even approaching the extinction of the system. This places considerable stress on the implemented algorithms as it is highly probable that, without appropriate controls, counts may become negative. Furthermore, this allows for the observation of system extinctions due to stochastic fluctuations rather than the deterministic model, which does not undergo extinction under any condition.

4.2 Computational times and error analysis

Simulations will be terminated upon reaching the predetermined maximum time. In order to assess the efficiency of the algorithms, the average computational times will be computed over 10^4 simulations, except for the Gillespie algorithm for $V = 10^6, 10^7, 10^8$, where the average will be based on 10^2 simulations. This approach is adopted to mitigate the prolonged computational duration that may arise while maintaining a statistical representativeness of the algorithm's performance in specific scenarios.

Regarding the error measure, in the theory of stochastic processes, it is not straightforward to define error, as there is no uniquely correct simulation, and each simulation from the same choice of method and parameters can yield different results. In this context, errors related to the expectations of processes are considered, with the widely used concept of *strong approximation error*: denoting X^V as the exact process simulated using the Gillespie method and Z^V as the approximated process simulated using τ-leaping, this error is defined as:

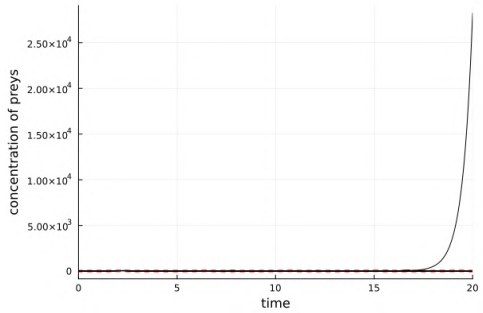
$$\sup_{t \leq T} \mathbb{E}||X^V(t) - Z^V(t)||. \tag{4.5}$$

In the simulations, the choice was made to use the Euclidean norm, but this selection is not crucial, as the discussion aims only to make comparisons between norms and not to use them in an absolute sense.

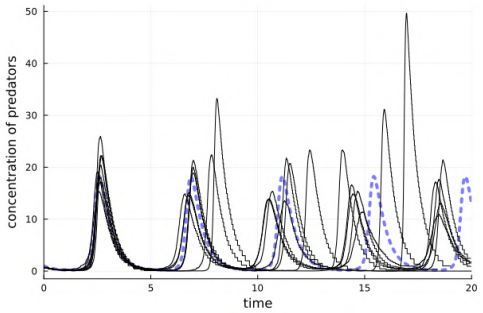
For each parameter choice and each approximated algorithm, the values of the chain at times 2.5, 5.0, 7.5, and 10.0 were recorded in each simulations. Through Monte Carlo approximation, the expected value and, consequently, the strong approximation error were estimated.

4.3 Application of the Gillespie algorithm

The Gillespie algorithm 2.1 is applied to the Lotka-Volterra model 4.2 just introduced for various orders of the volumetric term V. The results are presented below. The deterministic solutions of 4.1 for prey and predator concentrations are indicated in red and blue, respectively, while the 8 stochastic simulations obtained through the examined method are represented in black.

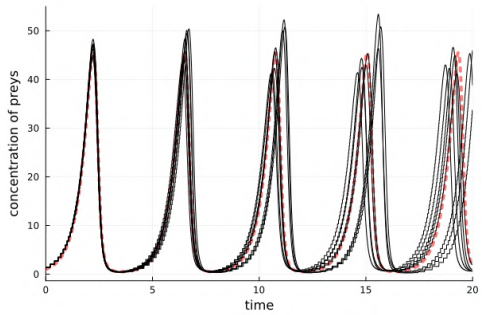


(a) Preys.

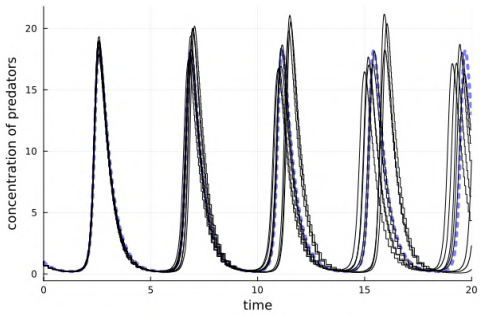


(b) Predators.

Figure 4.1: Simulations of the Lotka-Volterra model 4.2 by Gillespie algorithm 2.1, $V = 10^2$.

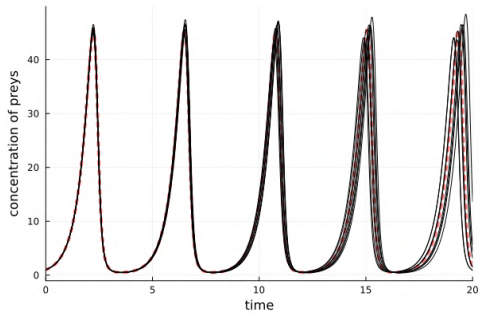


(a) Preys.

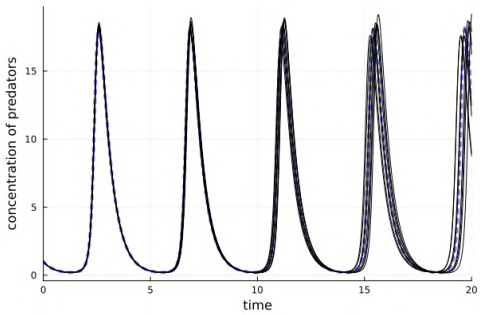


(b) Predators.

Figure 4.2: Simulations of the Lotka-Volterra model 4.2 by Gillespie algorithm 2.1, $V = 10^3$.

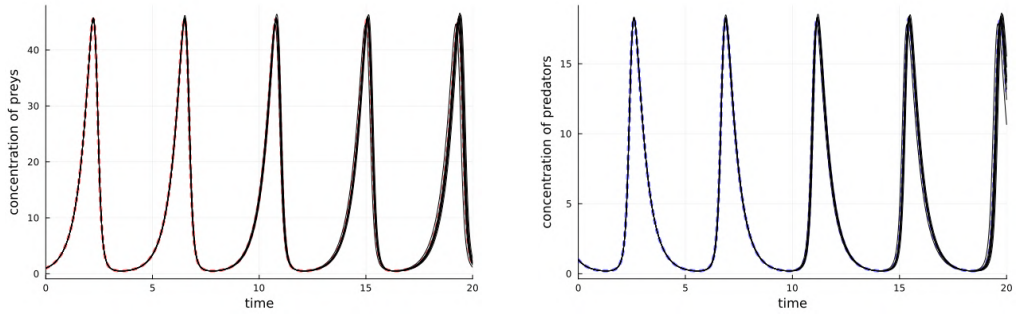


(a) Preys.



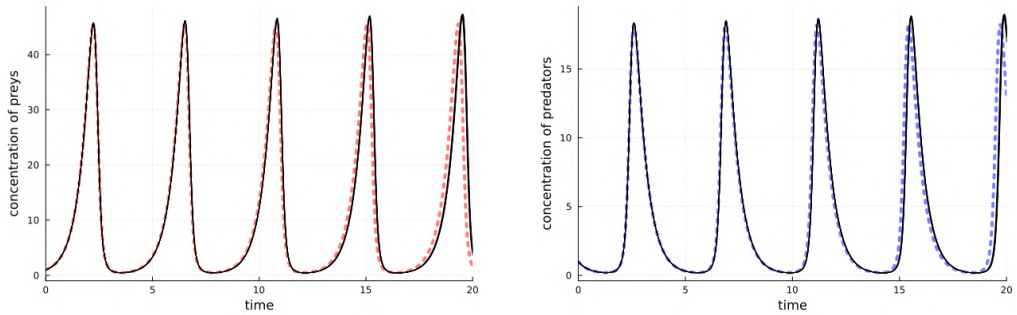
(b) Predators.

Figure 4.3: Simulations of the Lotka-Volterra model 4.2 by Gillespie algorithm 2.1, $V = 10^4$.



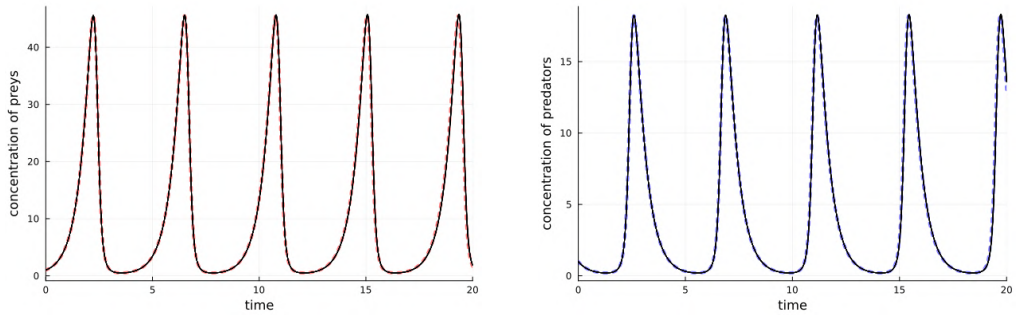
(a) Preys. (b) Predators.

Figure 4.4: Simulations of the Lotka-Volterra model 4.2 by Gillespie algorithm 2.1, $V = 10^5$.



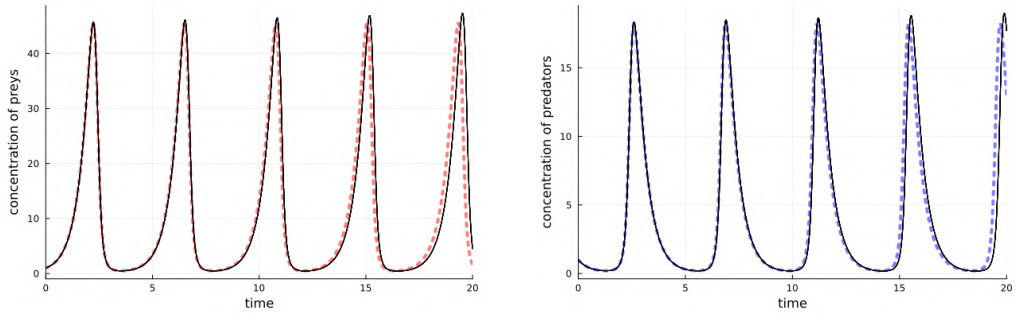
(a) Preys. (b) Predators.

Figure 4.5: Simulations of the Lotka-Volterra model 4.2 by Gillespie algorithm 2.1, $V = 10^6$.



(a) Preys. (b) Predators.

Figure 4.6: Simulations of the Lotka-Volterra model 4.2 by Gillespie algorithm 2.1, $V = 10^7$.



(a) Preys. (b) Predators.

Figure 4.7: Simulations of the Lotka-Volterra model 4.2 by Gillespie algorithm 2.1, $V = 10^8$.

Note that in Figure 4.1, multiple simulations reach extinction, meaning a null number of prey and predators. In [21], it is demonstrated that a system like the one under consideration almost certainly reaches extinction, and the estimated time for this to happen scales with the volumetric term. Therefore, it is justified that by increasing this term while keeping the observation time window constant, extinctions of the system were no longer observed.

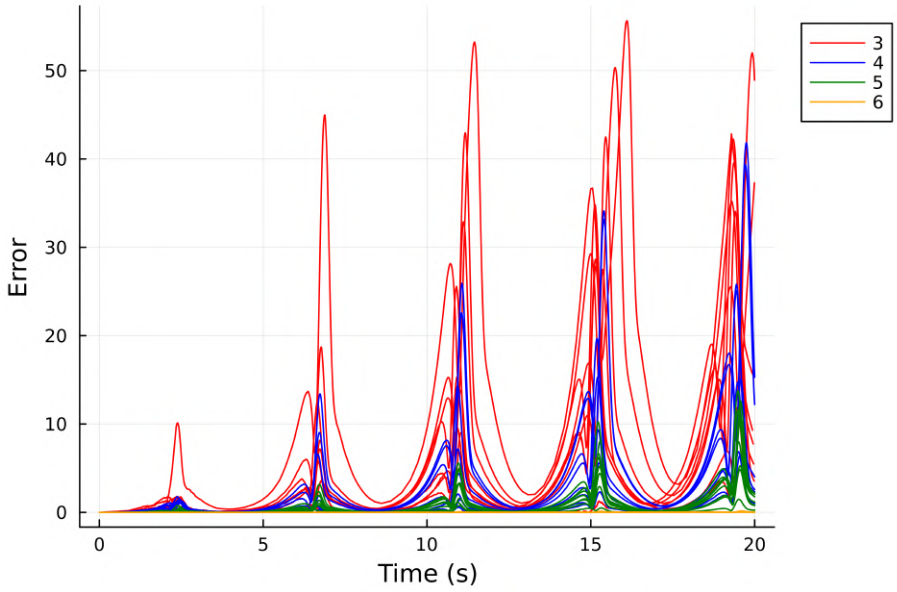


Figure 4.8: Absolute error in Euclidean norm of Gillespie algorithm simulations with respect to the deterministic solution for $V = 10^3, 10^4, 10^5, 10^6$.

The absolute error of the Gillespie algorithm simulations in Euclidean norm with respect to the deterministic solution tends to increase over time. This is easily explainable

by the stochastic nature of the simulations and the increasing uncertainty as time moves away from the initial moment at $t = 0$. In particular, the algorithm produces error spikes when the distributions of prey and predators also exhibit spikes: this is due to the fact that we are discussing about the absolute error, not the relative one that is not defined in two dimensions.

It is then observed that the increase in the volumetric term V leads to a significant reduction in the error, effectively demonstrating a convergence of stochastic simulations to the deterministic solution as $V \to \infty$. This convergence stabilizes the examined method as it satisfies the hypothesis of theorem 1.3.2.

The average computational times for $V = 10, 10^2, \dots, 10^8$ are reported. These values are averaged over 10^4 simulations for $V = 10^2, 10^3, 10^4, 10^5$, while they are averaged over 10^2 simulations for $V = 10^6, 10^7$, and over 10 simulations for $V = 10^8$ in order to avoid simulation times on the order of weeks.

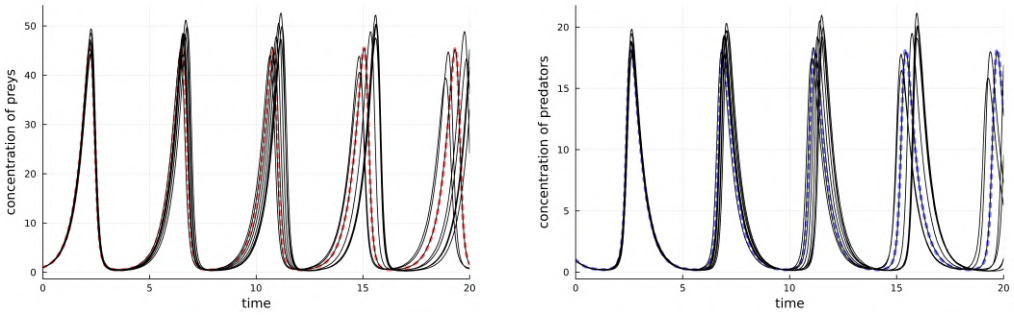
Gillespie	
V	Time (s)
10^2	0.045679700
10^3	0.379188065
10^4	3.565226987
10^5	36.24783459
10^6	374.4950714
10^7	3781.803827
10^8	38931.06881

Table 4.1: Average computational times of the Gillespie simulations on the Lotka-Volterra model 4.2 for $V = 10^2, \dots, 10^8$.

The increase in computational time exhibits a linear relationship with the variable V, resulting in a significant increase in computational costs for high orders of magnitude. This phenomenon compromises the validity of simulations, making them hardly distinguishable in terms of time from the conduct of a physical experiment, when applicable. Simulations for $V = 10$, $V = 10^8, 10^9, \dots$ are not reported due to the fact that computational times were so long that an simulation end was never observed.

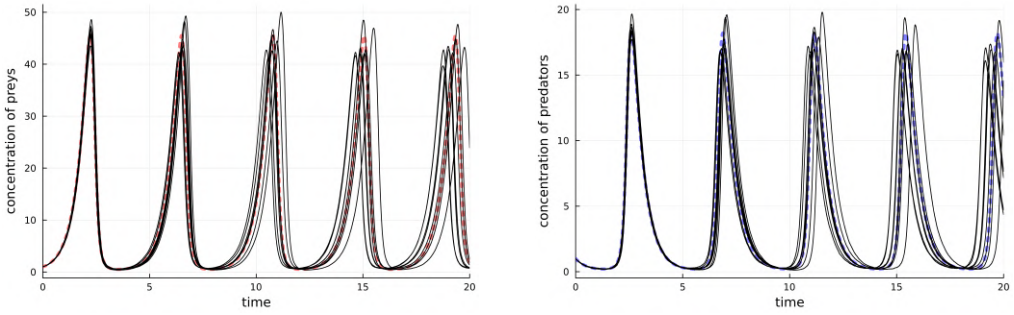
4.4 Application of the τ-leaping algorithm with fixed time interval

The τ-leaping algorithm with fixed time interval 3.1 is applied to the Lotka-Volterra model 4.2. $\tau = 10^{-3}$ was chosen as higher values of τ often resulted in simulation failures. The results for various values of the volumetric term V are presented below.



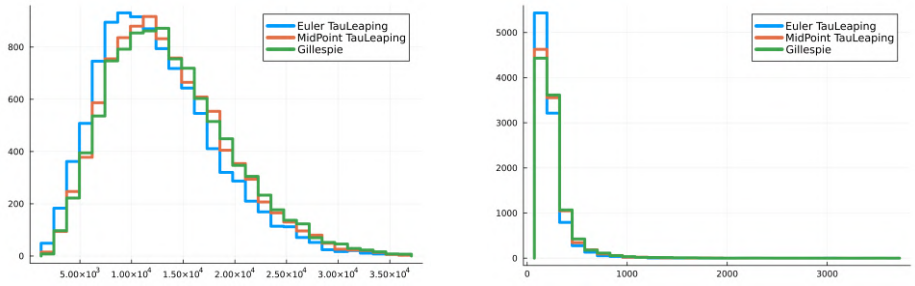
(a) Preys. (b) Predators.

Figure 4.9: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$, $V = 10^3$.



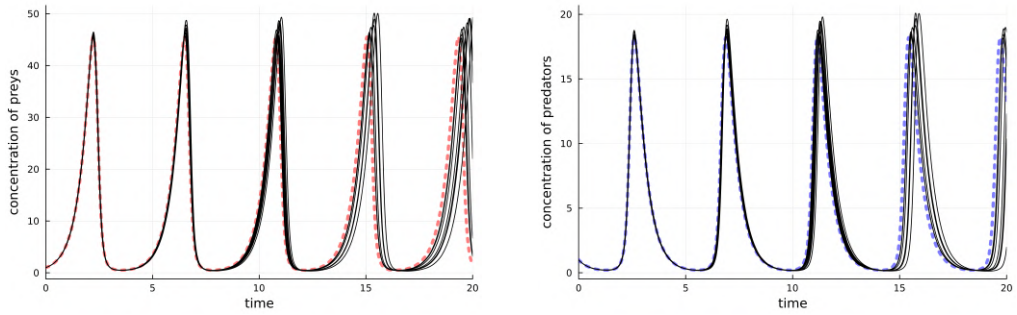
(a) Preys. (b) Predators.

Figure 4.10: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$ and MidPoint correction, $V = 10^3$.



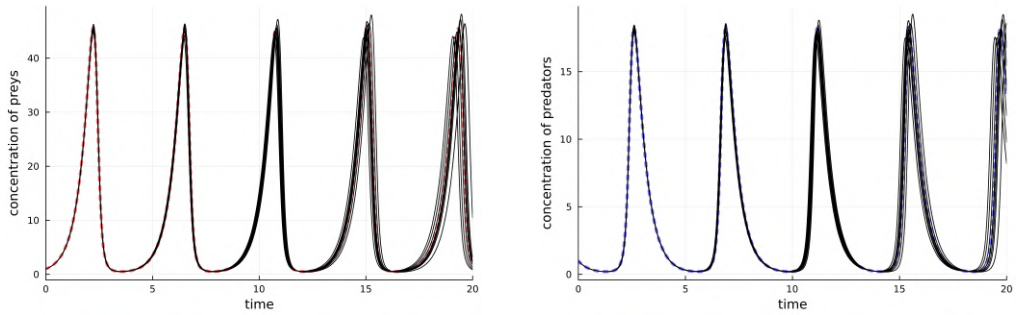
(a) Preys. (b) Predators.

Figure 4.11: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with fixed time interval with $\tau = 10^{-3}$, MidPoint τ-leaping with fixed time interval with $\tau = 10^{-3}$, $V = 10^3$.



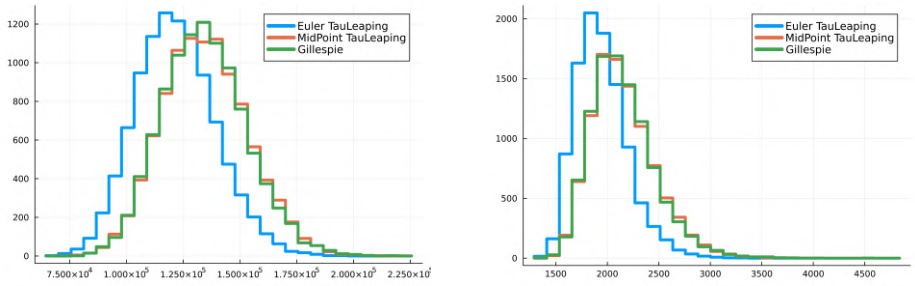
(a) Preys. (b) Predators.

Figure 4.12: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$, $V = 10^4$.



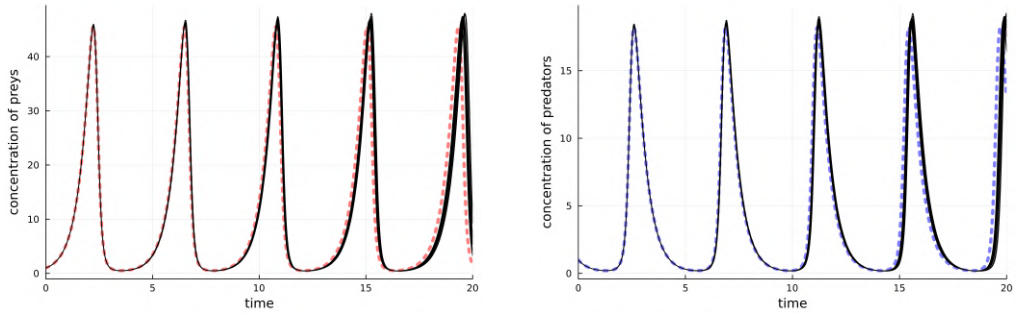
(a) Preys. (b) Predators.

Figure 4.13: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$ and MidPoint correction, $V = 10^4$.



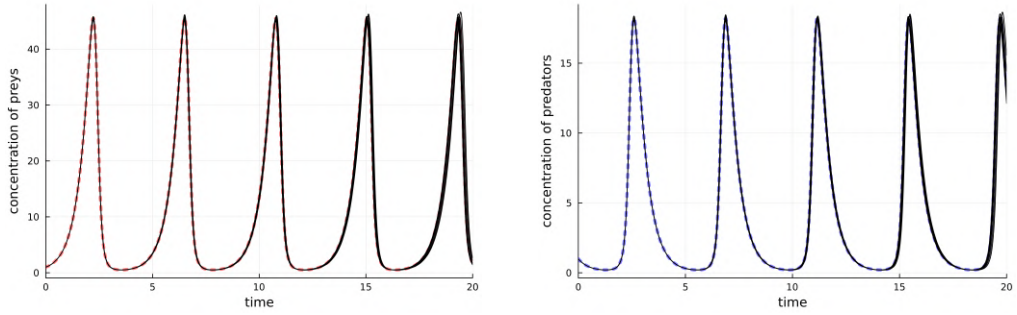
(a) Preys. (b) Predators.

Figure 4.14: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with fixed time interval with $\tau = 10^{-3}$, MidPoint τ-leaping with fixed time interval with $\tau = 10^{-3}$, $V = 10^4$.



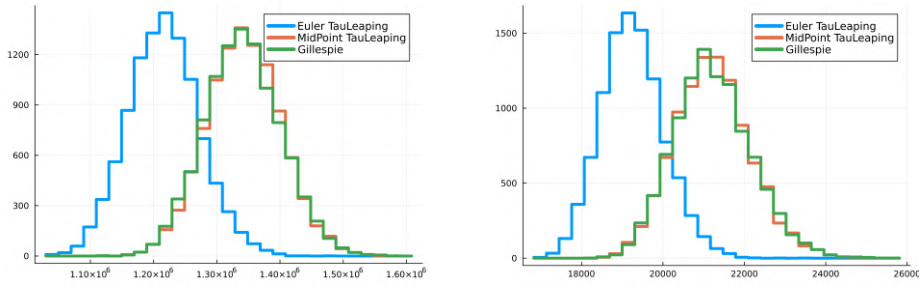
(a) Preys. (b) Predators.

Figure 4.15: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$, $V = 10^5$.



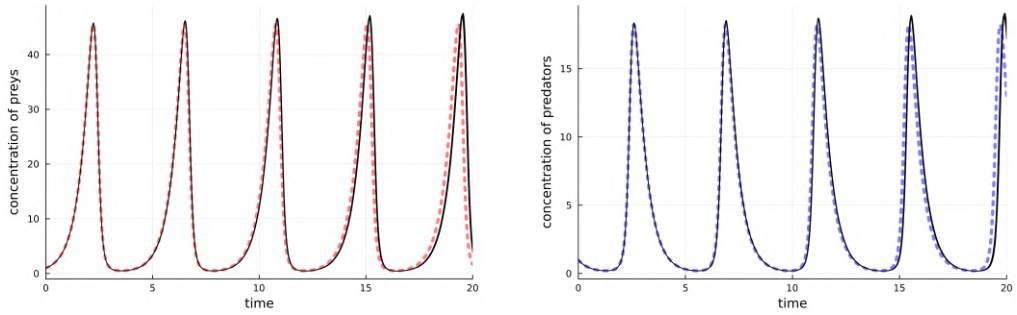
(a) Preys. (b) Predators.

Figure 4.16: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$ and MidPoint correction, $V = 10^5$.



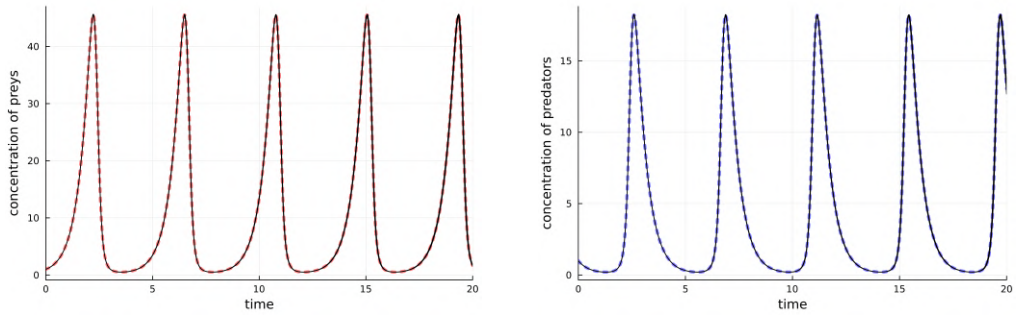
(a) Preys. (b) Predators.

Figure 4.17: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with fixed time interval with $\tau = 10^{-3}$, MidPoint τ-leaping with fixed time interval with $\tau = 10^{-3}$, $V = 10^5$.



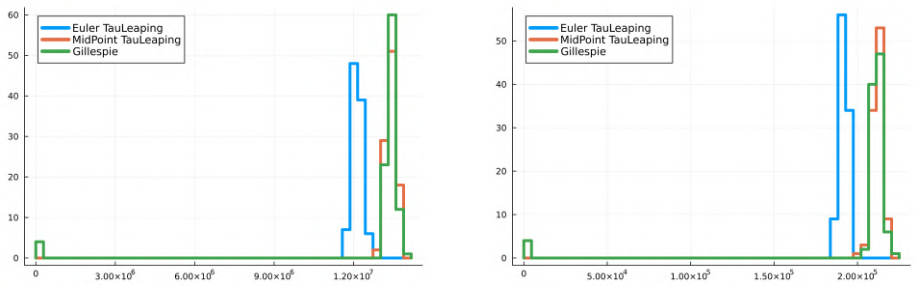
(a) Preys. (b) Predators.

Figure 4.18: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$, $V = 10^6$.



(a) Preys. (b) Predators.

Figure 4.19: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-3}$ and MidPoint correction, $V = 10^6$.



(a) Preys. (b) Predators.

Figure 4.20: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with fixed time interval with $\tau = 10^{-3}$, MidPoint τ-leaping with fixed time interval with $\tau = 10^{-3}$, $V = 10^6$.

It is observed that, especially for high values of V, the empirical distribution of $X(10.0)$ obtained by τ-leaping with a fixed time interval and MidPoint correction is closer to the empirical distribution obtained by Gillespie, with respect to the one without MidPoint correction.

The average computational times and the strong approximation errors for $t = 2.5$, 5.0, 7.5, 10.0 for $V = 10{,}10^2, \ldots, 10^8$ are reported. The term *fail* refers to combinations of parameters that led to negative population values with high frequency.

τ-leaping with fixed time interval τ - Euler					
V	Time(s)	Error 2.5	Error 5.0	Error 7.5	Error 10.0
10	fail	fail	fail	fail	fail
10^2	fail	fail	fail	fail	fail
10^3	0.01465353	3.461051e0	8.269607e-1	2.275599e0	6.539465e0
10^4	0,01303461	1.087444e0	2.704530e-1	6.918767e-1	2.275183e0
10^5	0.01604792	4.014836e-1	1.292330e-1	3.508865e-1	1.295386e0
10^6	0.01583959	2.551641e-1	1.211298e-1	3.371656e-1	1.286854e0
10^7	0.01408372	2.682280e-1	1.196079e-1	3.293710e-1	1.233607e0
10^8	0.01547784	2.565012e-1	1.164233e-1	3.257434e-1	1.247894e0

Table 4.2: Average computational times and strong approximation errors for $t = 2.5, 5.0, 7.5, 10.0$ for $V = 10{,}10^2, \ldots, 10^8$ over 10^4 simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with fixed time interval 3.1 $\tau = 10^{-3}$.

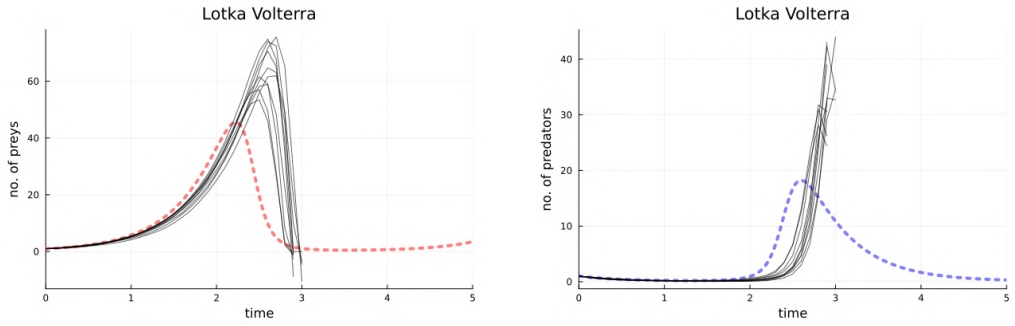
τ-leaping with fixed time interval τ - MidPoint					
V	Time(s)	Error 2.5	Error 5.0	Error 7.5	Error 10.0
10	fail	fail	fail	fail	fail
10^2	fail	fail	fail	fail	fail
10^3	0.01362476	3.393591e0	8.057146e-1	2.1237710e0	6.512625e0
10^4	0,01443733	1.056784e0	2.565367e-1	6.200809e-1	2.107579e0
10^5	0.01447899	3.526097e-1	1.969130e-1	1.969130e-1	6.731810e-1
10^6	0.01419684	1.463284e-1	6.847297e-2	6.847297e-2	2.255723e-1
10^7	0.01370139	1.268111e-1	8.475225e-3	2.389180e-2	1.098159e-1
10^8	0.01348307	1.292627e-1	1.402189e-3	2.997006e-3	1.288219e-2

Table 4.3: Average computational times and strong approximation errors for $t = 2.5, 5.0, 7.5, 10.0$ for $V = 10{,}10^2, \ldots, 10^8$ over 10^4 simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with fixed time interval 3.1 $\tau = 10^{-3}$ and MidPoint correction.

It is evident that the computational times are much lower compared to those observed in simulations using the Gillespie algorithm, and they do not scale with the volumetric term V. This is crucial when choosing approximate algorithms over exact ones when simulating at high cardinalities.

It is also noteworthy that the MidPoint correction has computational costs similar to the standard model. Regarding strong approximation errors, it is observed that not only is the upper limit on $t = 2.5, 5.0, 7.5$, and 10.0 for the MidPoint correction lower compared to the standard, but the error is overall lower point-wise.

A simulation for $\tau = 10^{-1}$ and $V = 10^2$ is shown.



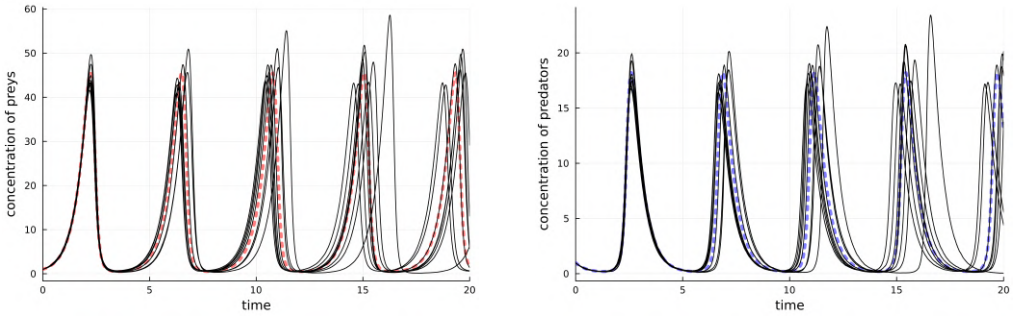
(a) Preys. (b) Predators.

Figure 4.21: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with fixed time interval $\tau = 10^{-1}$, $V = 10^2$.

One can observe how the 8 simulations of the prey concentration have taken negative values, leading to a halt in the algorithm. This signifies that a choice of τ that is excessively high, especially at low orders of magnitude of the volumetric term, results in a failure of the τ-leaping method. This is attributed to the fact that the regular τ-leaping algorithm does not verify the physical feasibility of the number of jumps.

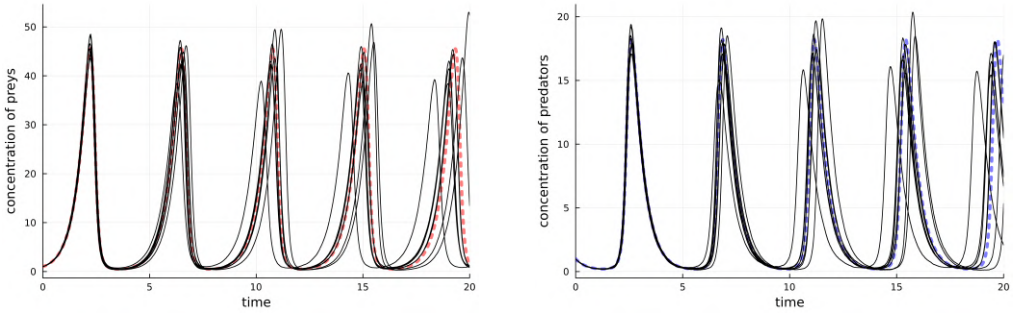
4.5 Application of the τ-leaping algorithm with adaptive time interval

The τ-leaping algorithm 3.1 with adaptive time step 3.6 is applied to the Lotka-Volterra model 4.2. A value of $\varepsilon = 10^{-2}$ has been chosen. The results for various values of the volumetric term V are presented below.



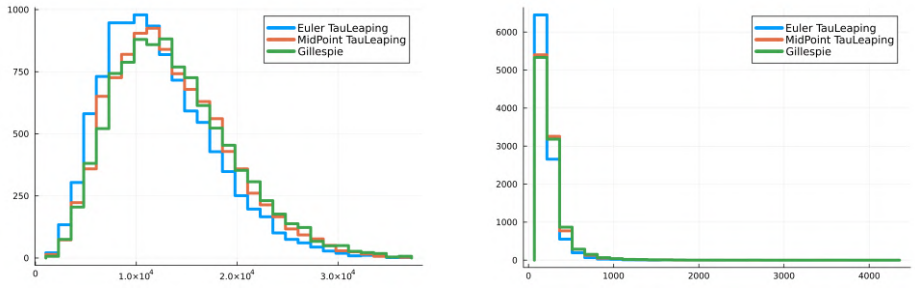
(a) Preys. (b) Predators.

Figure 4.22: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^3$.



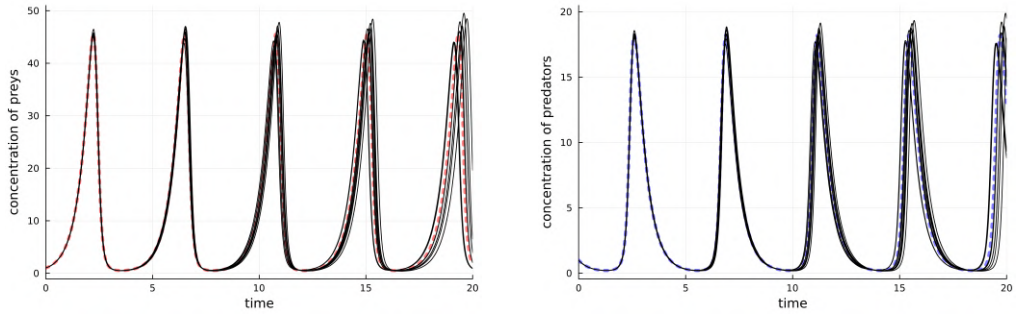
(a) Preys. (b) Predators.

Figure 4.23: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval and MidPoint correction, $\varepsilon = 10^{-2}$, $V = 10^3$.



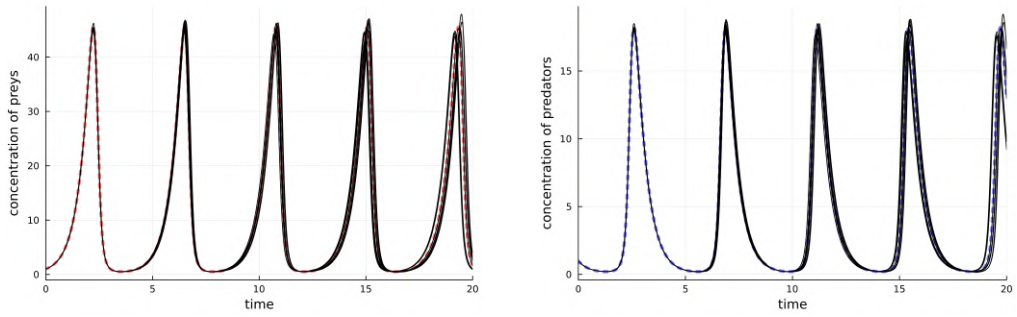
(a) Preys. (b) Predators.

Figure 4.24: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with adaptive time interval, MidPoint τ-leaping with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^6$.



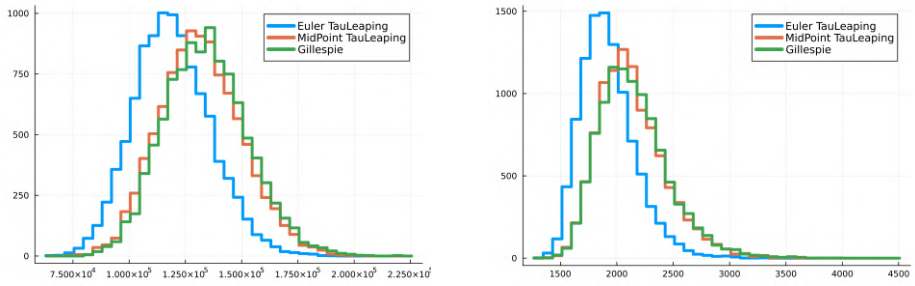
(a) Preys. (b) Predators.

Figure 4.25: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^4$.



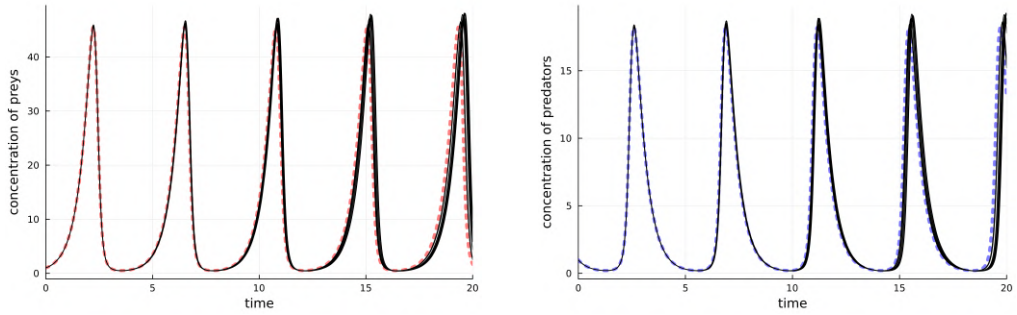
(a) Preys. (b) Predators.

Figure 4.26: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval and MidPoint correction, $\varepsilon = 10^{-2}$, $V = 10^4$.



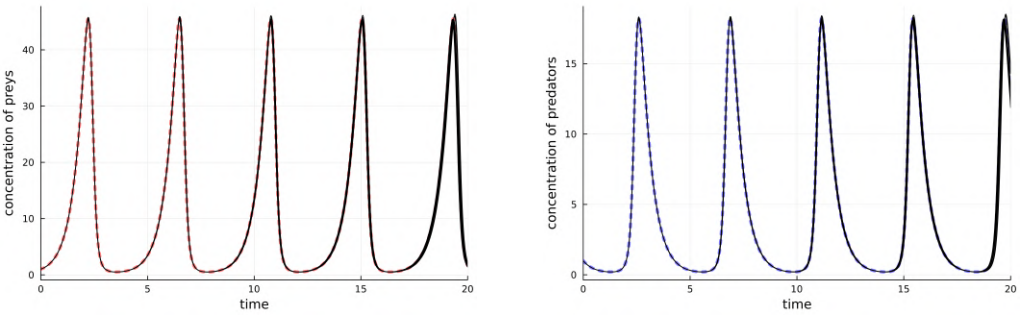
(a) Preys. (b) Predators.

Figure 4.27: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with adaptive time interval, MidPoint τ-leaping with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^4$.



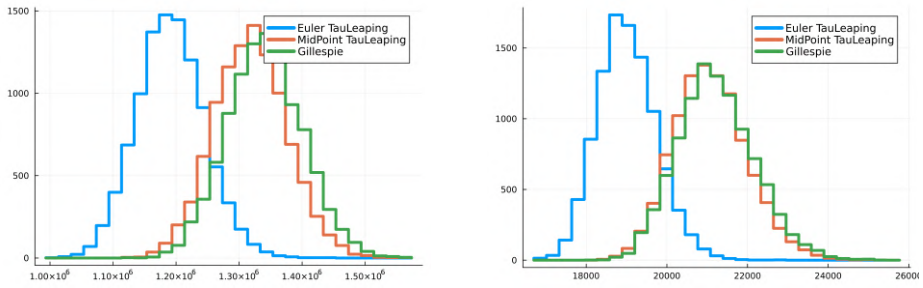
(a) Preys. (b) Predators.

Figure 4.28: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^5$.



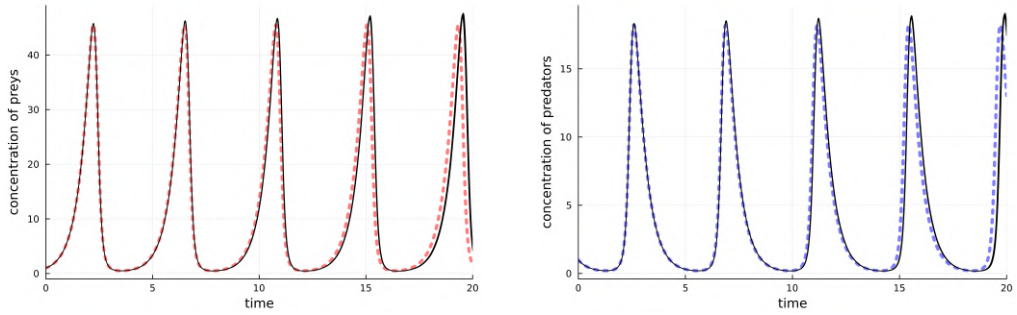
(a) Preys. (b) Predators.

Figure 4.29: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval and MidPoint correction, $\varepsilon = 10^{-2}$, $V = 10^5$.



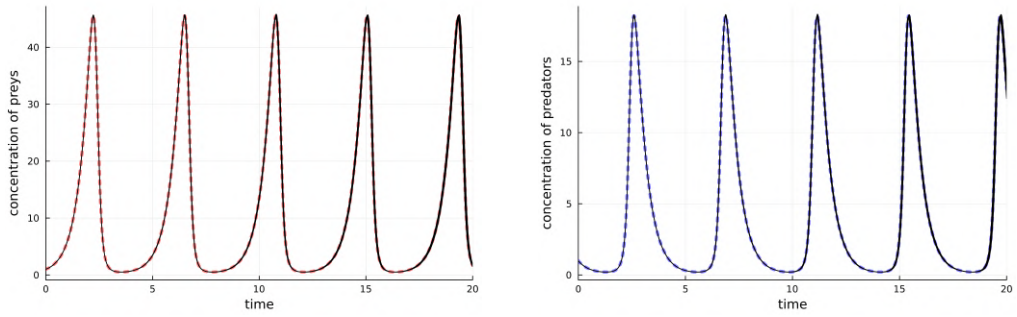
(a) Preys. (b) Predators.

Figure 4.30: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with adaptive time interval, MidPoint τ-leaping with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^5$.



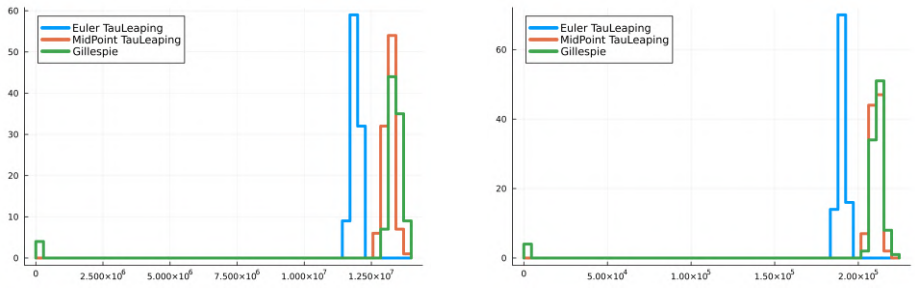
(a) Preys. (b) Predators.

Figure 4.31: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^6$.



(a) Preys. (b) Predators.

Figure 4.32: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm 3.1 with adaptive time interval and MidPoint correction, $\varepsilon = 10^{-2}$, $V = 10^6$.



(a) Preys. (b) Predators.

Figure 4.33: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with adaptive time interval, MidPoint τ-leaping with adaptive time interval, $\varepsilon = 10^{-2}$, $V = 10^6$.

The average computational times and the strong approximation errors for $t = 2.5$, 5.0, 7.5, 10.0 for $V = 10,10^2,\ldots,10^8$ are reported. The term *fail* refers to combinations of parameters that led to negative population values with high frequency.

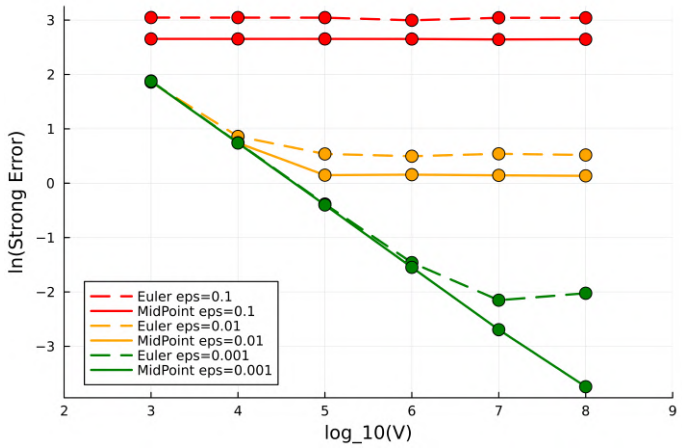
τ-leaping with adaptive time interval - Euler						
ε	V	Time(s)	Error 2.5	Error 5.0	Error 7.5	Error 10.0
0.1	10	fail	fail	fail	fail	fail
	10^2	fail	fail	fail	fail	fail
	10^3	0.003200257	2.103752e1	1.446424e0	7.536107e0	1.000413e1
	10^4	0.003021609	2.102419e1	1.392425e0	6.749721e0	9.644569e0
	10^5	0.004595978	2.100862e1	1.391601e0	6.689032e0	9.638542e0
	10^6	0.002546805	2.012689e1	1.336975e0	6.392649e0	9.249814e0
	10^7	0.002495197	2.095918e1	1.365134e0	6.671666e0	9.640396e0
	10^8	0.002675172	2.094016e1	1.358010e0	6.744318e0	9.610727e0
0.01	10	fail	fail	fail	fail	fail
	10^2	fail	fail	fail	fail	fail
	10^3	0.027805727	3.765126e0	8.166304e-1	2.253756e0	6.426305e0
	10^4	0.026921315	1.472067e0	2.871497e-1	7.474322e-1	2.363945e0
	10^5	0.026535232	1.712616e0	1.715337e-1	4.518428e-1	1.516631e0
	10^6	0.025641159	1.169923e0	1.611955e-1	4.260573-1	1.443968e0
	10^7	0.022165885	1.717408e0	1.640898e-1	4.405615e-1	1.454628e0
	10^8	0.020437094	1.679761e0	1.652254e-1	4.345715e-1	1.469610e0
0.001	10	fail	fail	fail	fail	fail
	10^2	fail	fail	fail	fail	fail
	10^3	0.203382034	3.425626e0	8.223270e-1	2.167364e0	6.554215e0
	10^4	0.280973714	1.066236e1	2.569309e-2	6.237021e-1	2.106967e0
	10^5	0.240279234	3.365998e-1	8.306790e-2	1.999507e-1	6.814724e-1
	10^6	0.234305969	9.549171e-2	2.598128e-2	6.259181e-2	2.321038e-1
	10^7	0.261152711	4.654829e-2	1.254434e-2	3.078268e-2	1.162662e-1
	10^8	0.247715237	4.019459e-2	1.206370e-2	2.995567e-2	1.321693e-1

Table 4.4: Average computational times and strong approximation errors for $t = 2.5, 5.0, 7.5, 10.0$ for $V = 10,10^2,\ldots,10^8$ over 10^4 simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with adaptive time interval 3.6, $\varepsilon = 10^{-2}$.

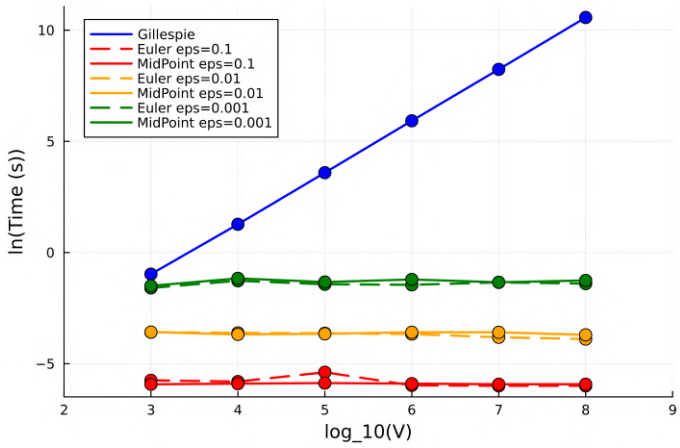
τ-leaping with adaptive time interval - MidPoint						
ε	V	Time(s)	Error 2.5	Error 5.0	Error 7.5	Error 10.0
0.1	10	fail	fail	fail	fail	fail
	10^2	fail	fail	fail	fail	fail
	10^3	0.002675172	1.421412e1	8.662844e-1	2.623644e0	6.287537e3
	10^4	0.002749225	1.420064e1	5.121107e-1	1.364222e0	2.601878e4
	10^5	0.002821282	1.420244e1	4.982743e-1	1.308117e0	2.117086e5
	10^6	0.002752515	1.418349e1	5.075525e-1	1.332186e0	2.152854e0
	10^7	0.002680988	1.406517e1	5.097661e-1	1.325454e0	2.120821e0

	10^8	0.002681654	1.410891e1	5.295950e-1	1.297305e0	2.235005e0
0.01	10	fail	fail	fail	fail	fail
	10^2	fail	fail	fail	fail	fail
	10^3	0.027984626	3.632804e0	8.123871e-1	2.253756e0	6.497984e0
	10^4	0.025224371	1.472067e0	2.601647e-1	6.383087e-1	2.098968e0
	10^5	0.025887036	1.159338e0	9.065453e-2	2.163132e-1	6.907420e-1
	10^6	0.027734828	1.169923e0	5.381461e-2	1.173005e-1	2.769983-1
	10^7	0.027654741	1.155430e0	4.582078e-2	1.003836e-1	1.940517e-1
	10^8	0,024704109	1.145194e0	4.878444e-2	9.598947e-2	2.297867e-1
0.001	10	fail	fail	fail	fail	fail
	10^2	fail	fail	fail	fail	fail
	10^3	0.224051787	3.372500e0	8.177261e-1	2.140431e0	6.518575e0
	10^4	0.313424703	1.057398e0	2.553920e-1	6.174357e-1	2.094241e0
	10^5	0.265579655	3.363535e-1	8.133263e-2	1.954903e-1	6.701169e-1
	10^6	0.298377177	1.059487-1	2.542036e-2	6.037334e-2	2.127593e-1
	10^7	0,262956461	4.439571e-2	8.836932e-3	2.125995e-2	6.763700e-2
	10^8	0,285288223	1.367134e-2	1.983025e-3	6.113851e-3	2.369861e-2

Table 4.5: Average computational times and strong approximation errors for $t = 2.5, 5.0, 7.5, 10.0$ for $V = 10$,$10^2, \ldots,10^8$ over 10^4 simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with adaptive time interval 3.6 and MidPoint correction, $\varepsilon = 10^{-2}$.



(a) Strong Error Analysis.



(b) Time Analysis.

Figure 4.34: Analysis of the strong approximation error and average computational times of the τ-leaping method 3.1 with adaptive step size 3.6 using $\varepsilon = 0.1, 0.01, 0.001$ in relation to the order of the volumetric term V.

From the initial analysis of computational times, similarly to the case of a fixed time interval, no clear relationship with the volumetric term V emerges. Instead, an inverse proportionality with the control parameter ε is highlighted. This implies that relaxing the *leap condition* control leads to significantly faster simulations compared to an exact approach, such as that implemented with the Gillespie algorithm.

Furthermore, it is noteworthy that the implementation of the MidPoint correction does not significantly impact the method's performance, demonstrating a negligible increase in computational times. Additionally, the MidPoint correction leads to improvements in terms of strong approximation error in all cases. Therefore, when applying the τ-leaping algorithm, it is always advantageous to incorporate the MidPoint correction.

4.6 Application of the τ-leaping algorithm with Post-Leap Check

The τ-leaping algorithm with Post-Leap Check 3.2 is applied to the Lotka-Volterra model 4.2.

Since this model relies on the choice of four parameters ε, p, p^*, q, a *Grid Search* algorithm was applied to evaluate average computational times and strong approximation errors generated by the τ-leaping algorithm with Post-Leap Check for every possible combination of parameters (clearly excluding combinations where $p^* < p$). The choices where from with $\varepsilon \in \{0.001, 0.01, 0.1\}$, $p \in \{0.1, \ldots, 0.9\}$, $p^* \in \{0.1, \ldots, 0.9\}$, $q \in \{0.1, \ldots, 0.9\}$. This led to the selection of the following parameters:

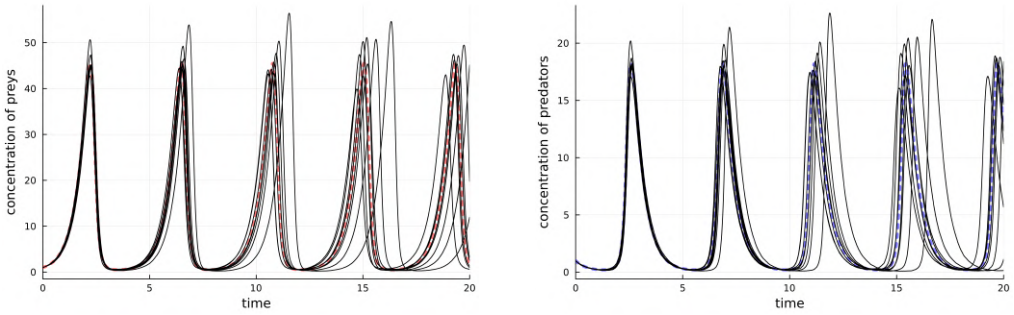
$$\varepsilon = 0.01; \quad p = 0.1; \quad p^* = 0.5; \quad q = 0.7$$

These were chosen in order to maintain reasonable computational times, albeit higher than the standard τ-leaping algorithm, yet yielding smaller strong approximation errors. Indeed, the control parameter ε being small enforces the validity of the *leap condition*, and the algorithm ensures that this condition is not violated, resulting in longer times due to the intrinsic matrix handling of the algorithm. However, it also leads to a simulation closer to the exact one.

The decision was made to use these parameters indiscriminately for every value of the volumetric term V, even though their selection is based solely on simulations with $V = 10^4$. This choice stems from the observation that computational times do not vary significantly across orders of magnitude of the volumetric term. Additionally, while errors decrease with an increase in V, the relative ordering of errors concerning other parameter choices remains relatively consistent.

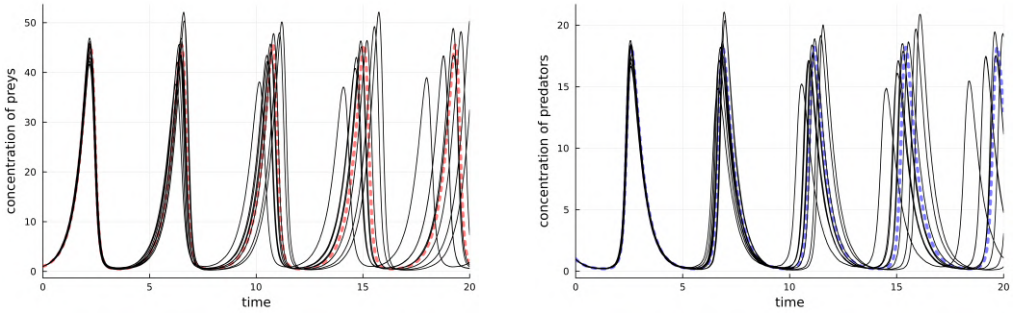
It has been observed that combinations of parameters involving $\varepsilon = 0.001$, high values of p, and low values of q result in computational times on the order of hours. Despite offering slight improvements in terms of strong approximation errors, these improvements do not justify the excessive increase in computational times.

The results for various values of the volumetric term V are presented below.



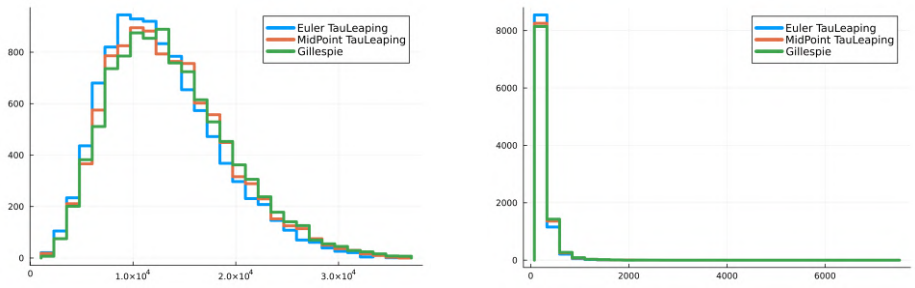
(a) Preys. (b) Predators.

Figure 4.35: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^3$.



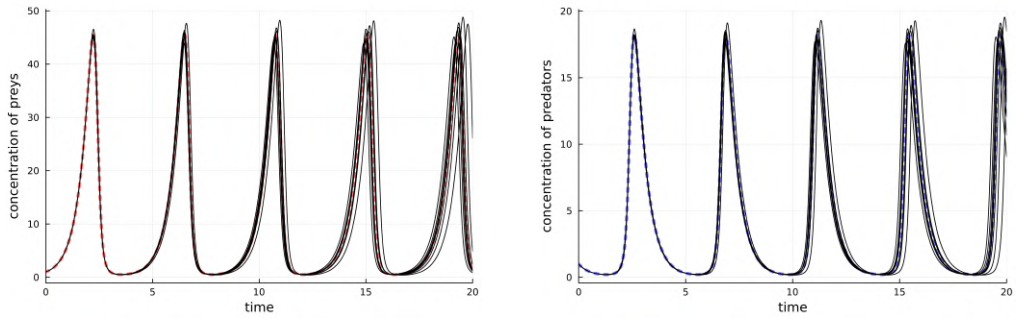
(a) Preys. (b) Predators.

Figure 4.36: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check and MidPoint correction, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^3$.



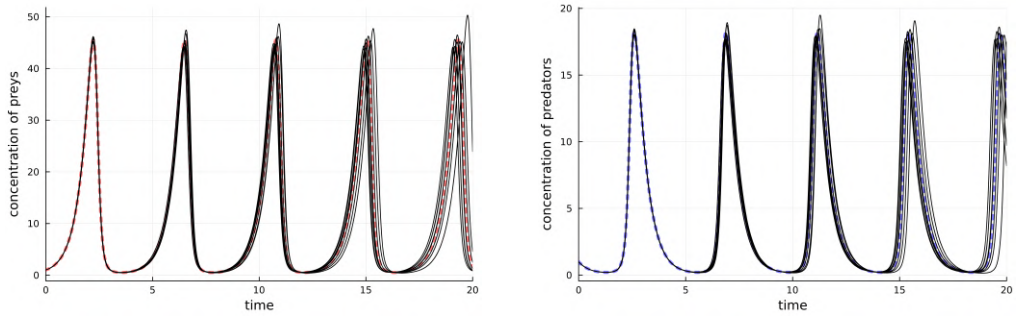
(a) Preys. (b) Predators.

Figure 4.37: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with Post-Leap Check, MidPoint τ-leaping with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^3$.



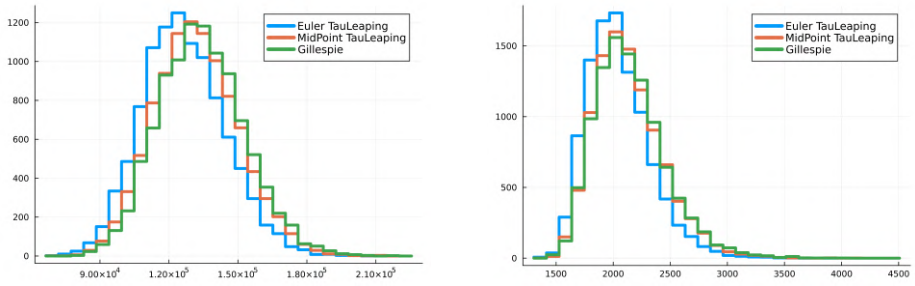
(a) Preys. (b) Predators.

Figure 4.38: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^4$.



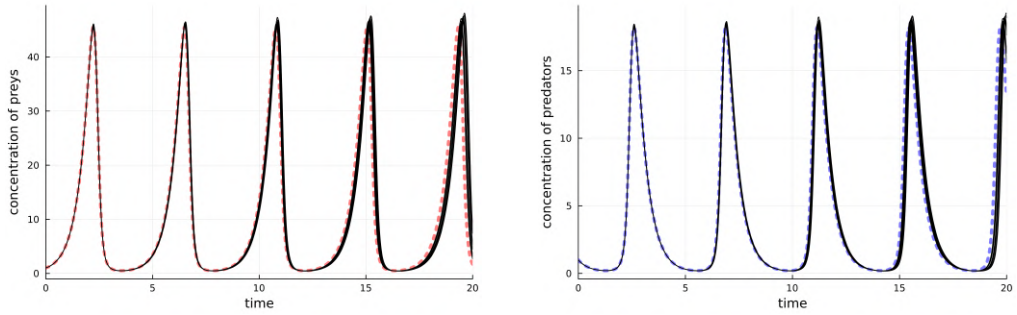
(a) Preys. (b) Predators.

Figure 4.39: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check and MidPoint correction, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^4$.



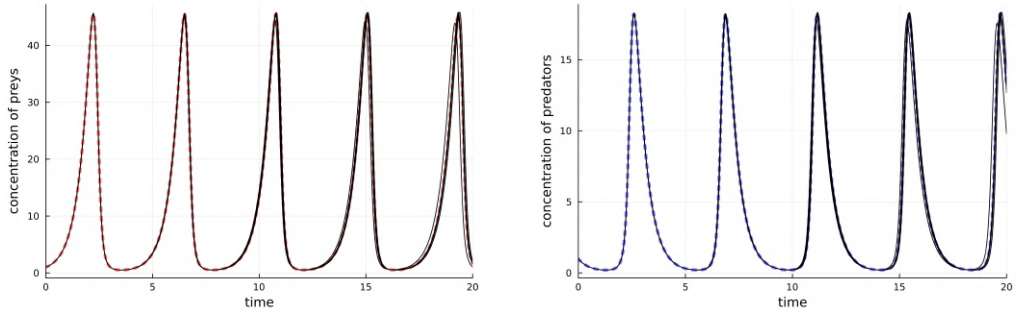
(a) Preys. (b) Predators.

Figure 4.40: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with Post-Leap Check 3.2, MidPoint τ-leaping with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^4$.



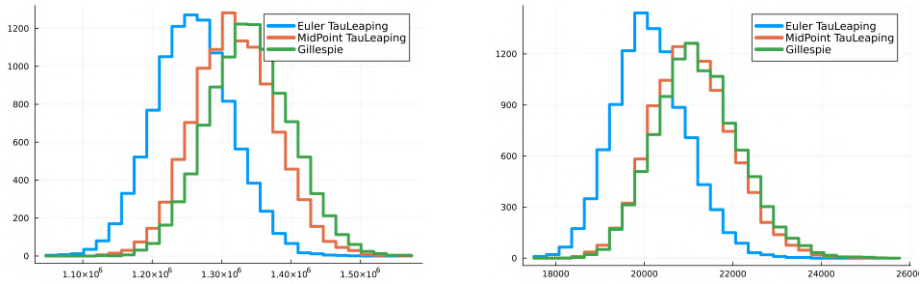
(a) Preys. (b) Predators.

Figure 4.41: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^5$.



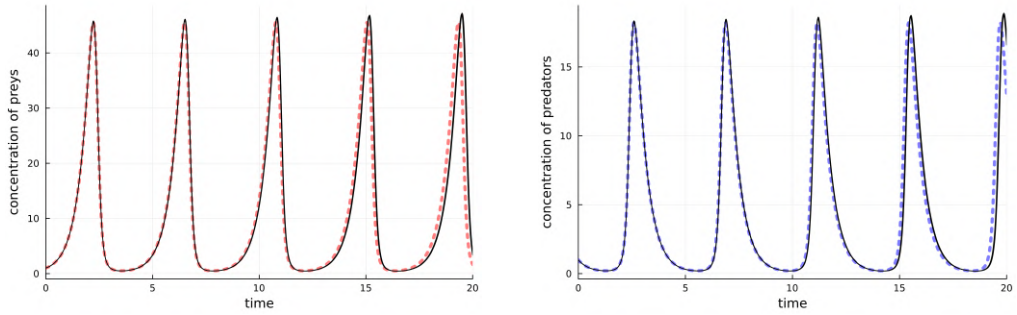
(a) Preys. (b) Predators.

Figure 4.42: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check and MidPoint correction, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^5$.



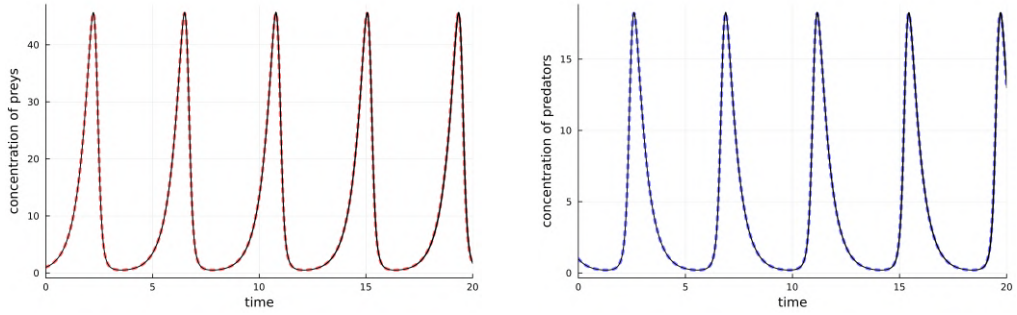
(a) Preys. (b) Predators.

Figure 4.43: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with Post-Leap Check 3.2, MidPoint τ-leaping with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^5$.



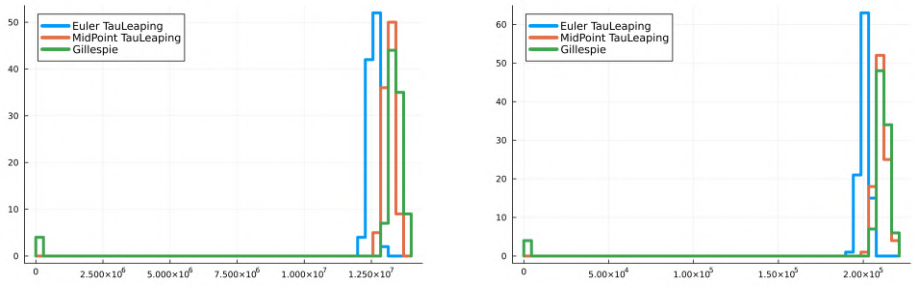
(a) Preys. (b) Predators.

Figure 4.44: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^6$.



(a) Preys. (b) Predators.

Figure 4.45: Simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check and MidPoint correction, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^6$.



(a) Preys. (b) Predators.

Figure 4.46: Occurrences of the number of preys and predators in $t = 10.0$ from 10^4 simulations of the Gillespie algorithm 2.1, Euler τ-leaping with Post-Leap Check 3.2, MidPoint τ-leaping with Post-Leap Check, $\varepsilon = 10^{-2}, p = 0.1, p^* = 0.5, q = 0.7$, $V = 10^6$.

The average computational times and the strong approximation errors for $t = 2.5$, 5.0, 7.5, 10.0 for $V = 10{,}10^2, \ldots, 10^8$ are reported. The term *fail* refers to combinations of parameters that led to negative population values with high frequency.

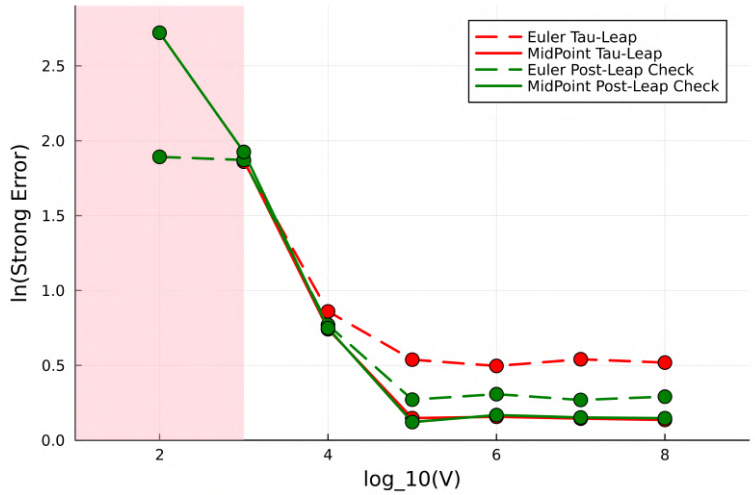
τ-leaping with Post-Leap Check - Euler					
V	Time(s)	Error 2.5	Error 5.0	Error 7.5	Error 10.0
10	slow	slow	slow	slow	slow
10^2	0.97296929	6.634224e0	1.991173e0	1.877346e0	1.081297e1
10^3	0.67334203	3.996273e0	8.467086e-1	2.340372e0	6.498901e0
10^4	0.57612039	1.573623e0	2.664989e-1	6.558222e-1	2.159355e0
10^5	0.64573776	1.311412e0	1.189543e-1	2.959635e-1	9.046285e-1
10^6	0.49575222	1.360226e0	2.384205e-1	5.621524e-1	1.294613e0
10^7	0.49477867	1.308779e0	1.075110e-1	2.681420e-1	7.913229e-1
10^8	0.49677709	1.337845e0	1.130042e-1	2.825580e-1	8.489548e-1

Table 4.6: Average computational times and strong approximation errors for $t = 2.5, 5.0, 7.5, 10.0$ for $V = 10{,}10^2, \ldots, 10^8$ over 10^4 simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check 3.2, $\varepsilon = 10^{-2}$, $p = 0.1$, $p^* = 0.5$, $q = 0.7$.

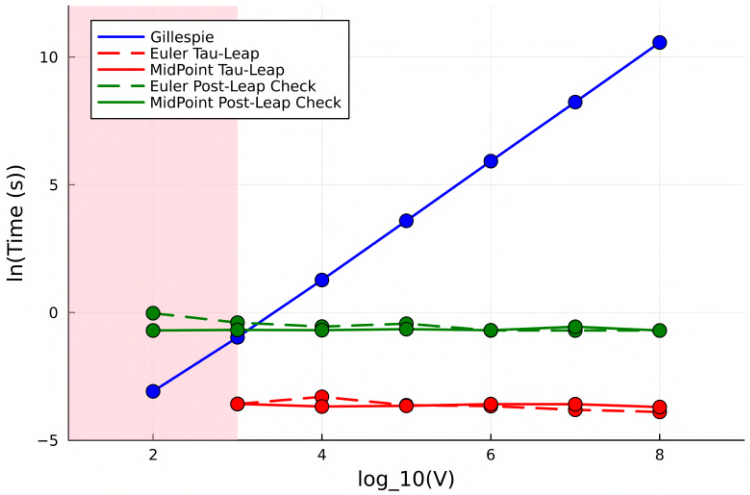
τ-leaping with Post-Leap Check - MidPoint					
V	Time(s)	Error 2.5	Error 5.0	Error 7.5	Error 10.0
10	slow	slow	slow	slow	slow
10^2	0.49485696	1.519825e1	2.156039e0	8.496327e0	1.197000e1
10^3	0.50541004	3.405076e0	7.514808e-1	1.927938e0	6.855707e0
10^4	0.50075953	1.803287e0	2.624639e-1	6.563208e-1	2.114483e0
10^5	0.52118702	1.128817e0	8.565933e-2	2.036075e-1	5.929277e-1
10^6	0.49915973	1.183073e0	1.907087e-1	4.159442e-1	7.840760e-1
10^7	0.57256692	1.164460e0	5.496103e-2	1.189239e-1	1.997243e-1
10^8	0.49456155	1.158743e0	5.348339e-2	1.167743e-1	2.084621e-1

Table 4.7: Average computational times and strong approximation errors for $t = 2.5, 5.0, 7.5, 10.0$ for $V = 10{,}10^2, \ldots, 10^8$ over 10^4 simulations of the Lotka-Volterra model 4.2 by τ-leaping algorithm with Post-Leap Check 3.2 and MidPoint correction, $\varepsilon = 10^{-2}$, $p = 0.1$, $p^* = 0.5$, $q = 0.7$.

The decision has been made to compare the τ-leaping model with Post-Leap Check using optimized parameters $\varepsilon = 0.01$, $p = 0.1$, $p^* = 0.5$, $q = 0.7$ against the τ-leaping method with the same control parameter $\varepsilon = 0.01$.



(a) Strong Error Analysis.



(b) Time Analysis.

Figure 4.47: Analysis of the strong approximation error and average computational times of the τ-leaping method with Post-Leap Check 3.2 using $\varepsilon = 10^{-2}$, $p = 0.1$, $p^* = 0.5$, $q = 0.7$ and of the τ-leaping method 3.1 with adaptive time interval 3.6 using $\varepsilon = 10^{-2}$ in relation to the order of the volumetric term V.

The pink area indicates the values of the volumetric term that brought to a failure of the τ-leap algorithm 3.1; in those cases the usage of the τ-leaping algorithm with Post-Leap Check was mandatory.

It is evident that the τ-leaping algorithm with Post-Leap Check brings about significant improvements in terms of strong approximation error compared to the τ-leaping algorithm with adaptive step size. Once again, the MidPoint correction is observed to

enhance the method in terms of strong approximation error without significantly affecting computational time, making it an optimal addition. However, it is also noted that there are no substantial differences between τ-leaping with MidPoint correction and τ-leaping with Post-Leap Check and MidPoint correction. This result is closely tied to the specific case studied and may not be generalized universally.

Regarding computational costs, however, a significant increase in terms of time is observed compared to the version 3.1 of the τ-leaping algorithm.

Chapter 5

Application to the case study of nucleation and growth of gold nanoparticles

5.1 Introduction to the case study

Colloidal gold is a colloidal suspension (particles ranging from 88 nm to $1\mu m$ in a continuous dispersing substance) of nano and sub-microparticles of gold in a fluid, typically water. Depending on the concentration of gold and the size of the particles, the colloid exhibits different colors and properties.

The synthesis of such a colloid is often conducted in a liquid through the reduction of chloroauric acid: the latter is dissolved in the chosen liquid, and while it is stirred, a reducing agent is added, causing the reduction of Au^{+3} ions to neutral gold atoms. When the solution becomes supersaturated and gold atoms precipitate to the bottom, they attach to existing ones, forming gold nanoparticles. The addition of stabilizing agents can interrupt the formation of larger-sized particles.

Numerous are the applications of colloidal gold:

- The bacterium *Bacillus cereus*, when coated with gold nanoparticles with a poly-L-lysine film and subsequently washed with nitric acid, not only survives but also has its surface positively charged. When placed in a humid environment, these bacteria absorb water, swell their membrane, and allow the passage of current. [22]
- Colloidal gold has proven to play a significant role in the therapy for *rheumatoid arthritis* [23] [24]: in dogs, introducing a gold bead implant near the arthritic hip joint considerably reduces pain. [25]
- A therapy involving colloidal gold and microwaves is currently under in vitro study to combat Alzheimer's disease: irradiating a region injected with gold nanoparticles

with microwaves could enable the destruction of beta-amyloid plaques. [26]

- The use of gold nanoparticles for *drug delivery* is well-known, such as in the transport of paclitaxel (taxol). Nanoparticle capsules have proven to be unrecognizable by the reticuloendothelial system, allowing drug absorption without being digested by macrophages. [27]
- There are numerous applications for the detection and ablation of cancerous tissues that are challenging to reach. [28]

5.2 Formulation of the model

A model is intended to be formulated to describe the aggregation of nanoparticles, where the objective is to monitor the various sizes formed and the number of nanoparticles for each size.

In [29], LaMer and Dinegar initially developed a model concerning the formation of monodisperse particles in hydrodispersions. A more comprehensive investigation focusing on gold nanoparticles is presented in [30]. From these studies and subsequent ones, it is apparent that, assuming the presence of L gold nanoparticle monomers, denoted as M, two processes are feasible:

1. **Nucleation**, that is m monomers aggregate to form a nanoparticle of size m, denoted as P_m;
2. **Growth**, that is a monomer M and a nanoparticle P_i aggregate in order to form a nanoparticle of size $i+1$, denoted as P_{i+1}.

The natural progression involves constructing a Chemical Reaction Network that models the processes of nucleation and growth of nanoparticles:

$$\begin{aligned} mM &\xrightarrow{\nu} P_m \\ M + P_i &\xrightarrow{\gamma} P_{i+1}, \quad i = m, \dots, L-1 \end{aligned} \tag{5.1}$$

Let X_0 represent the count of monomers, and X_i the count of nanoparticles of size i. From here, the nucleation rate and growth rates can be derived:

$$\begin{aligned} \lambda_0(X_0, X_1, \dots, X_L) &= \nu X_0(X_0-1)\dots(X_0-m+1); \\ \lambda_i(X_0, X_1, \dots, X_L) &= \gamma X_0 X_i; \quad i = m, \dots, L-1 \end{aligned}$$

and the reaction vectors:

$$\zeta_0 = (-m,0,\dots,0,1,0,\dots,0); \quad \zeta_i = (-1,0,\dots,0,-1,1,0,\dots,0).$$

In the environment of *Classical Scaling* we adopt:

$$\nu^V = \nu V^{1-m}; \quad \gamma^V = \frac{\gamma}{V}.$$

The model used for the simulations is determined by the following choice of constants: $\nu = 1.0$, $\gamma = 5.0$.

5.3 Application of the Gillespie algorithm in Classical Scaling

The Gillespie algorithm 2.1 is applied to the model 5.1 just introduced. Simulations are carried out until the depletion of monomers, as this results in the impossibility of any further reaction, leading to the system reaching a state of stasis.

These simulations are computationally demanding in terms of memory, as they require storing $\mathcal{O}(V)$ reaction rates. To mitigate this memory demand, it has been observed that simulations results in a high concentration of small-sized nanoparticles, albeit still smaller than the size $\lceil V^{\frac{1}{4}} \rceil$. Therefore, the storage has been restricted up to this size.

Regarding the reduction of computational times, it became evident that in the initial moments, there were not many sizes created, leading to null reaction rates. Therefore, starting from a null vector, only reaction rates involving present sizes were computed, significantly expediting the computation. Additionally, operations on integers were consistently prioritized before multiplication by rational constants to minimize computational times.

The results for various values of the volumetric term V are presented below.

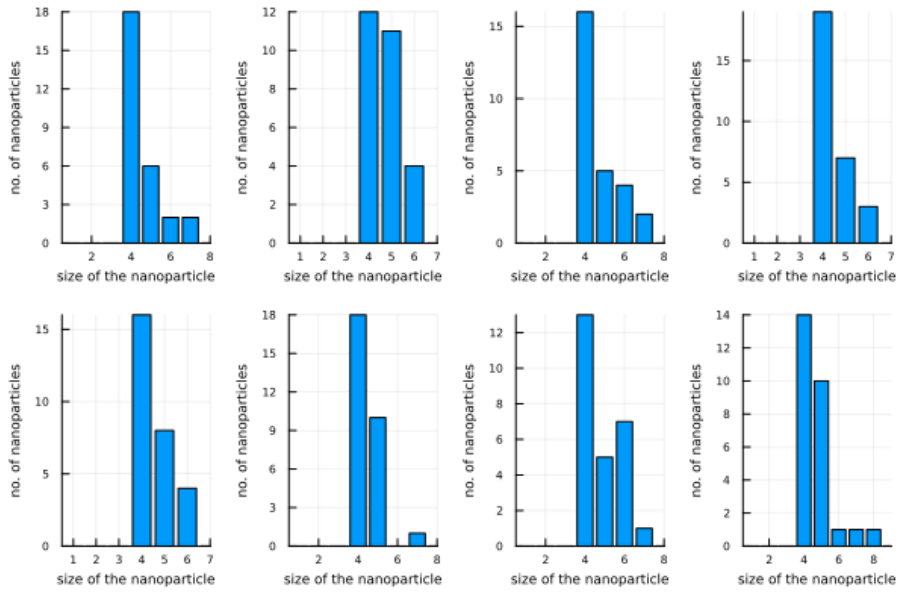


Figure 5.1: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Classical Scaling, $V = 10^2$.

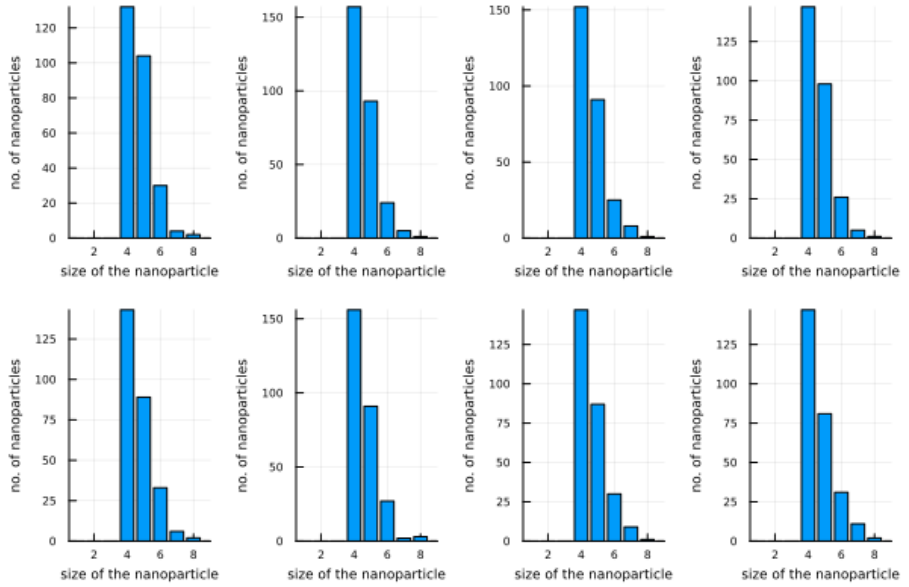


Figure 5.2: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Classical Scaling, $V = 10^3$.

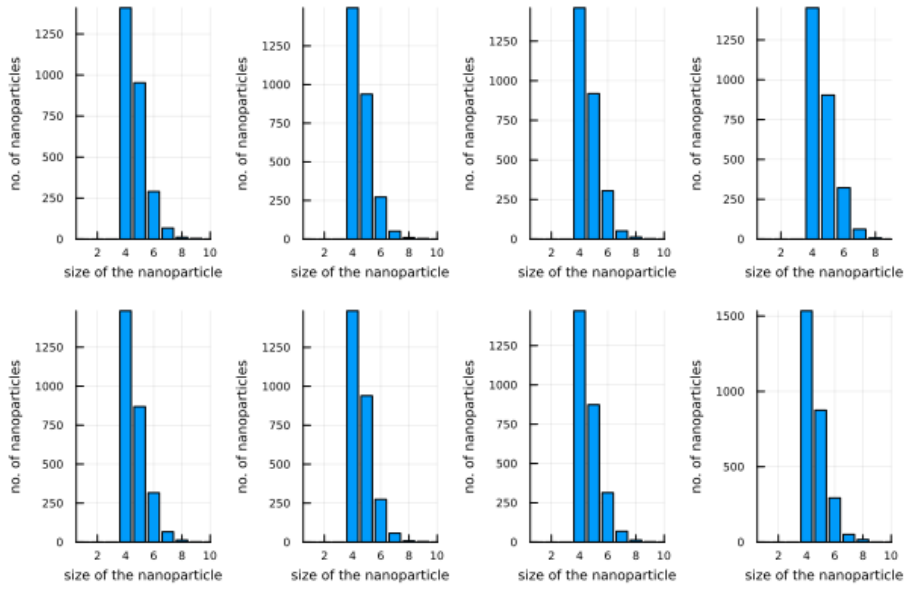


Figure 5.3: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Classical Scaling, $V = 10^4$.

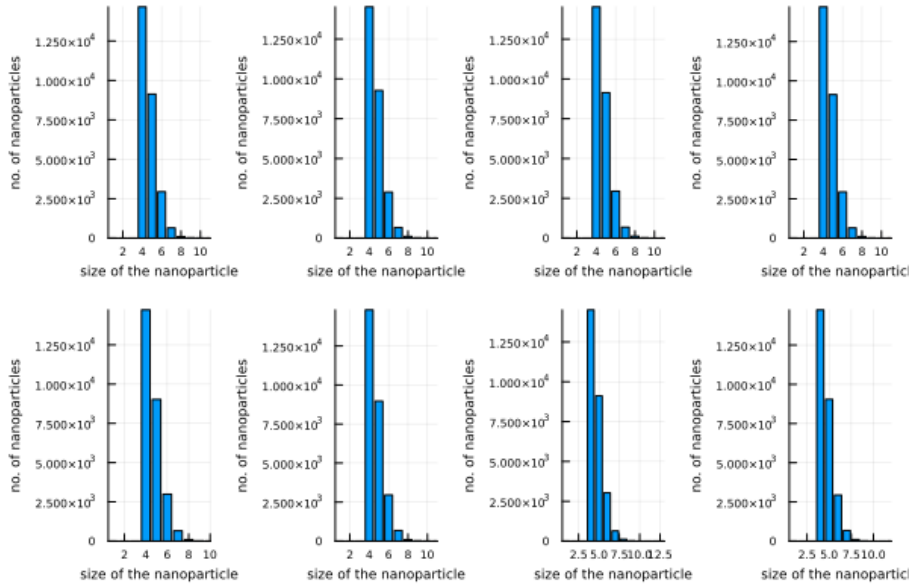


Figure 5.4: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Classical Scaling, $V = 10^5$.

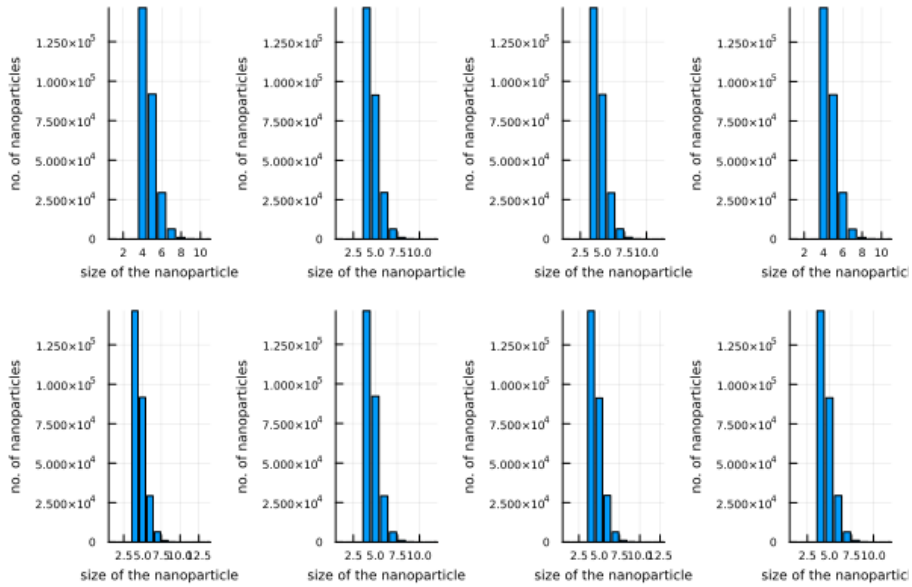


Figure 5.5: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Classical Scaling, $V = 10^6$.

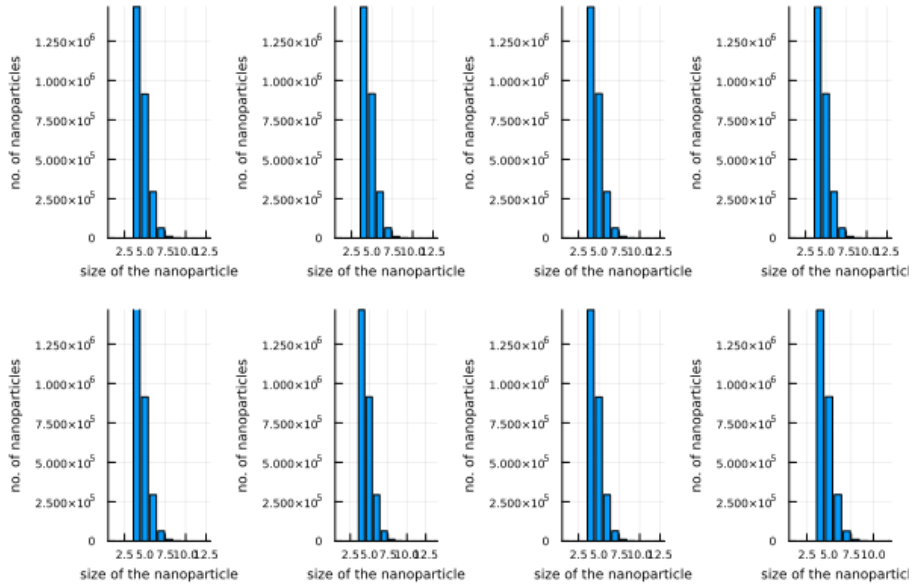


Figure 5.6: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Classical Scaling, $V = 10^7$.

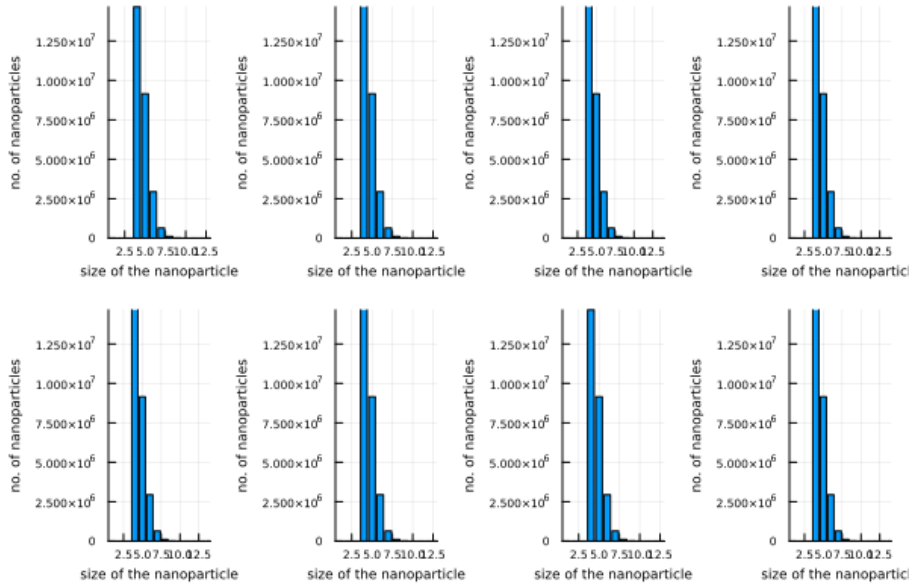


Figure 5.7: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Classical Scaling, $V = 10^8$.

In the initial phases, where the concentration of monomers is high and that of nanoparticles is low or zero, the probability of nucleation is of the order of $\mathcal{O}(V)$, while that of

growth is $\mathcal{O}(1)$. Therefore, in classical scaling, a high concentration of small-sized nanoparticles is obtained because, initially, many more nucleation reactions take place than growth reactions. Specifically, there were nanoparticles up to size 10 for almost all the simulations, with the highest frequency detected for nanoparticles of size 3. The number of nanoparticles decreases with its size.

The average computational times are reported for $V = 10, 10^2, \ldots, 10^8$: these values are averaged over 10^2 simulations for $V = 10, 10^2, 10^3, 10^4, 10^5, 10^6$, while they are averaged over 8 simulations for $V = 10^7, 10^8$ to avoid simulation times on the order of weeks.

Gillespie	
V	Time (s)
10	0.000549565
10^2	0.000479183
10^3	0.002146539
10^4	0.018651545
10^5	0.165757263
10^6	1.917495600
10^7	22.73620908
10^8	221.6518369

Table 5.1: Average computational times over 10^2 simulations of the model 5.1 in Classical Scaling by Gillespie algorithm 2.1 for various orders of the volumetric term V.

A clear increase in computational costs is observed with the increase in the volumetric term. These costs become substantial from $V = 10^7$, necessitating the implementation of approximate methods.

5.4 Application of the τ-leaping algorithm with Post-Leap Check in Classical Scaling

The application of the τ-leaping method 3.1, both with a fixed and adaptive time interval, leads to failure as the chain takes negative values. This is attributed to the limited presence of nanoparticles in the initial moments, often resulting in leap calculations assuming a greater number of nanoparticles. Therefore, the implementation of the τ-leaping algorithm with Post-Leap Check 3.2 becomes necessary, even though the failure of the τ-leaping algorithm predicts a high leap rejection frequency. This not only slows down simulations but also necessitates the allocation of a high number of rows for the matrices S_k.

Once again, the cost and memory reductions used for the Gillespie algorithm have been applied, with additional attention given to excluding elements related to nucleation from matrices and vectors. Nucleation elements, being frequently accessed, are saved in

separate variables.

Given that this model relies on the choice of four parameters ε, p, p^*, q, a *Grid Search* algorithm has been applied to evaluate average computational times and strong approximation errors generated by the τ-leaping algorithm with Post-Leap Check for every possible combination of parameters (clearly excluding combinations where $p^* < p$). The Grid Search algorithm was utilized with volumetric term $V = 10^5$ since choosing lower values would let the results be affected by stochastic fluctuations. Subsequent simulations were not conducted for different values of V. This decision was based on the observation that the parameters obtained for $V = 10^5$showed minimal deviation in both average computational times and approximation errors with respect to the optimal for other values of V.

The choices were from $\varepsilon \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, $p \in \{0.1, \ldots, 0.9\}$, $p^* \in \{0.1, \ldots, 0.9\}$, $q \in \{0.1, \ldots, 0.9\}$. Lower values of ε were not considered due to resulting in extremely long computational times. The following combinations have been considered:

- $\varepsilon = 0.5;\ p = 0.4;\ p^* = 0.6;\ q = 0.7$ that produces the minimum strong approximation error: in particular, with an error of 0.000299450353 and average computational time of 1.55548255 s;

- $\varepsilon = 0.9;\ p = 0.5;\ p^* = 0.6;\ q = 0.9$ that produces the minimum average computational time: in particular, with an error of 0.001125269015 and average computational time of 0.27303400 s;

- $\varepsilon = 0.6;\ p = 0.3;\ p^* = 0.4;\ q = 0.7$ that represents the trade-off minimum between average computational time and strong approximation error: in particular, with an error of 0.0003077198 and average computational time of 0.82081245 s.

Given the minimal difference in strong approximation error but the significant difference in computational time, the combination chosen for application is:

$$\varepsilon = 0.6; \quad p = 0.3; \quad p^* = 0.4; \quad q = 0.7.$$

Additionally, it's evident that the primary factor influencing the process is the control parameter ε. Lower values of ε lead to quicker computations but less precise approximations, while higher values have the opposite effect. In contrast, variables such as p, p^*, and q have a minimal impact on computation time. Therefore, a strategic approach to speeding up the Grid Search algorithm involves first assessing the significance of ε and then focusing on optimizing p, p^*, and q while maintaining ε at a fixed value.

The results for various values of the volumetric term V are presented below.

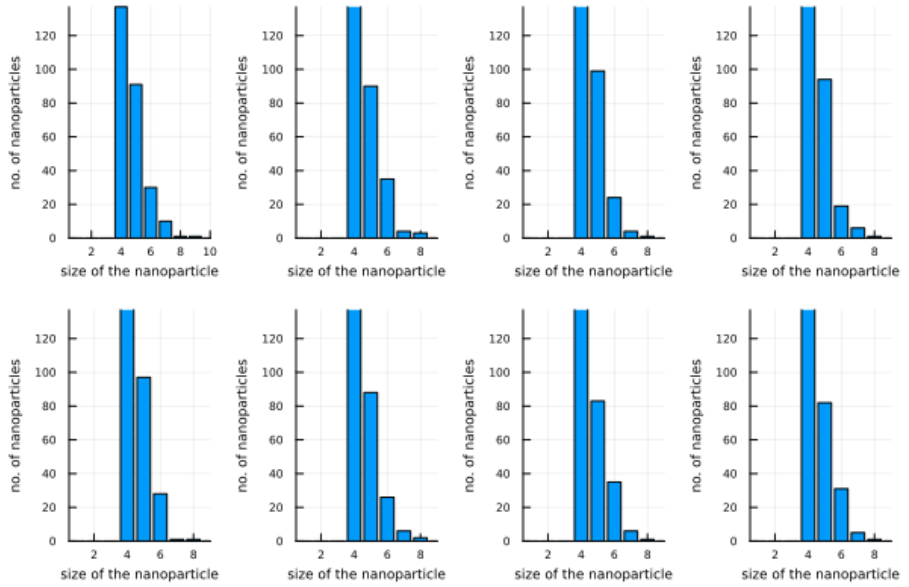


Figure 5.8: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Classical Scaling, $V = 10^3$.

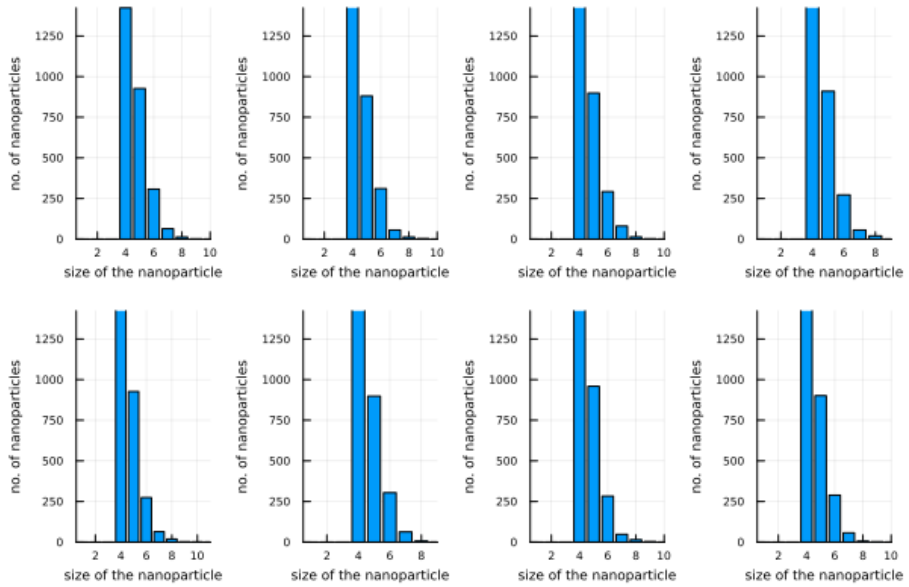


Figure 5.9: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Classical Scaling, $V = 10^4$.

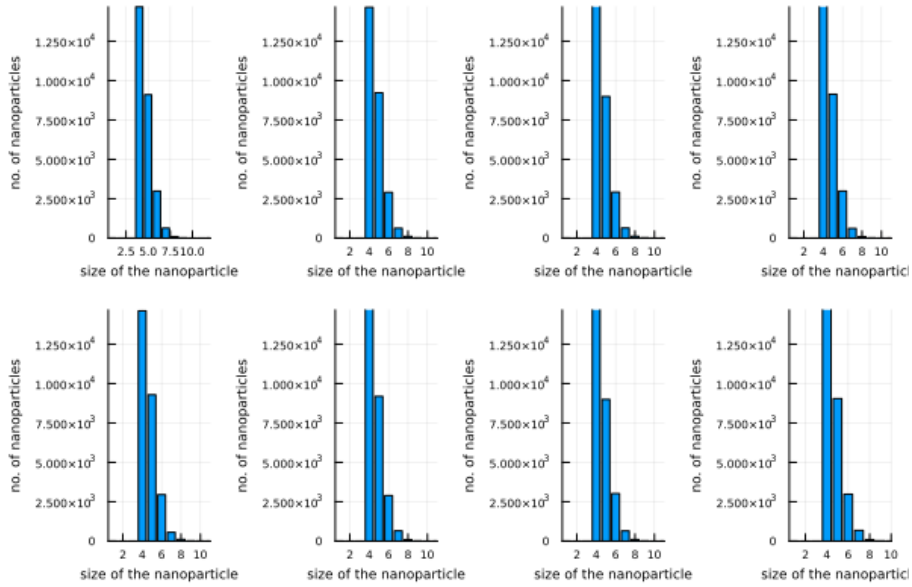


Figure 5.10: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Classical Scaling, $V = 10^5$.

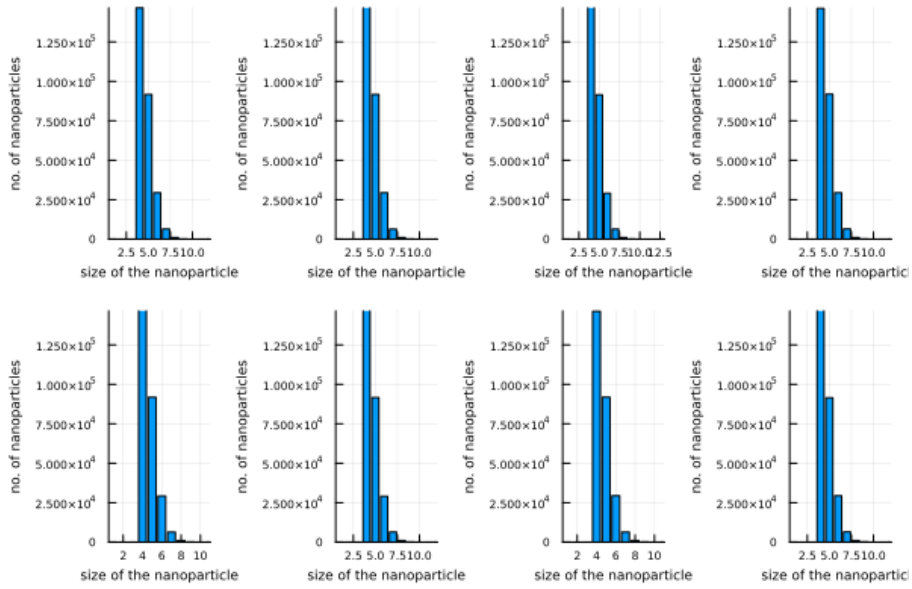


Figure 5.11: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Classical Scaling, $V = 10^6$.

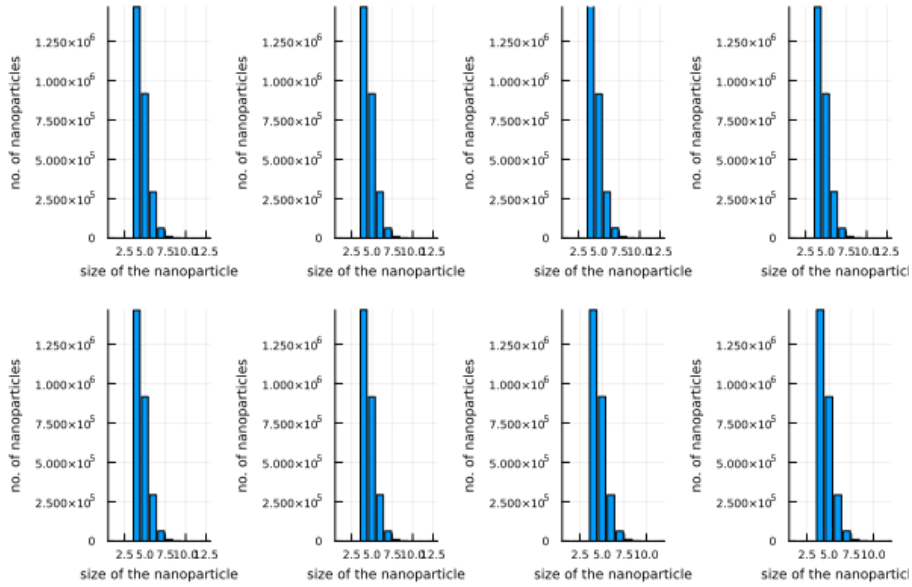


Figure 5.12: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Classical Scaling, $V = 10^7$.

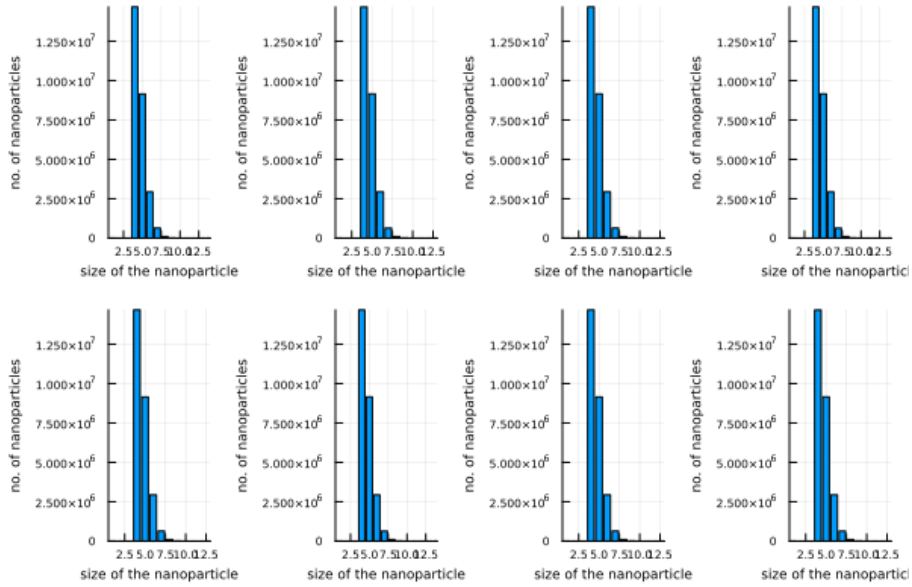


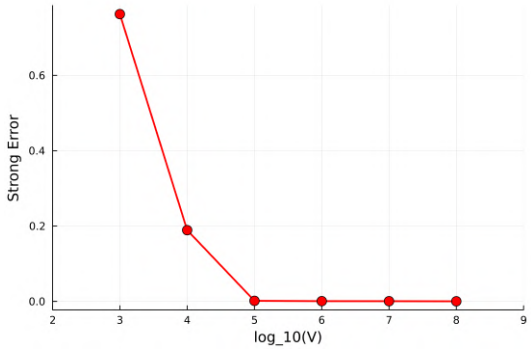
Figure 5.13: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Classical Scaling, $V = 10^8$.

The strong approximation error must be calculated over times that can maximize it. The error is been measured at $20\%, 40\%, 60\%, 80\%$ approximately of the process, in order

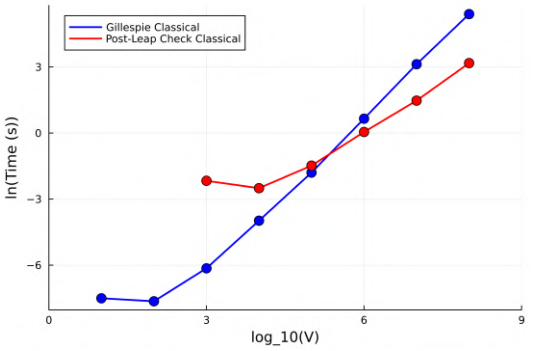
to obtain a larger error. The numerical results are presented below.

τ-leaping con Post-Leap Check - Classical					
V	Time(s)	Error 1	Error 2	Error 3	Error 4
10	slow	slow	slow	slow	slow
10^2	slow	slow	slow	slow	slow
10^3	0.11413710	7.625987e-1	5.913493e-1	4.049775e-1	2.150800e-1
10^4	0.08159760	1.890966e-1	1.782911e-1	1.764731e-1	1.760963e-1
10^5	0.22753717	1.183649e-3	6.882457e-4	1.154328e-3	1.442629e-3
10^6	1.04771119	1.390977e-4	3.503079e-4	4.860772e-4	4.948556e-4
10^7	4.35526904	8.320920e-5	2.131956e-4	3.325669e-4	4.948556e-4
10^8	23.9310644	4.548330e-5	5.649034e-5	1.905854e-4	8.434429e-5

Table 5.2: Average computational times and strong approximation errors for $V = 10,10^2,\ldots,10^8$ over 10^2 simulations of model 5.1 in Classical Scaling by τ-leaping with Post-Leap Check 3.2.



(a) Strong Error Analysis.



(b) Time Analysis.

Figure 5.14: Analysis of the strong approximation error and average computational times of the simulations of model 5.1 in Classical Scaling by the τ-leaping method with Post-Leap Check 3.2 using $\varepsilon = 0.6$, $p = 0.3$, $p^* = 0.4$, $q = 0.7$ in relation to the order of the volumetric term V.

The simulation results for $V = 10, 10^2$ using the τ-leaping algorithm with Post-Leap Check are not reported as the simulations never produced results due to exceedingly high computational times.

Looking at plot 5.14, it is easily observable that the error, as expected, decreases with the volumetric term. This is in agreement with theorem 1.3.2. Meanwhile, from the computational time graph, it is evident that the Gillespie algorithm is faster than the τ-leaping algorithm with Post-Leap Check up to $V = 10^5$, beyond which it is more advantageous to use the approximate method.

It has also been decided not to test the MidPoint correction on this case study since it presupposes the resolution, at least numerically, of the differential equations of the deterministic problem. As previously discussed, the analytical treatment of this problem is potentially impossible.

It's of particular interest the distribution of the different nanoparticles in size at the end

of the process: therefore, the expected errors in Euclidean norm of the final distribution obtained via τ-leaping method with Post-Leap Checks with respect to the one obtained via Gillespie method are computed.

Expect errors	
V	
10	slow
10^2	slow
10^3	1.913321e-2
10^4	5.305929e-3
10^5	1.649143e-3
10^6	5.671941e-4
10^7	2.656721e-4
10^8	1.810236e-4

Table 5.3: Expected errors of simulations of the model 5.1 by τ-leaping method with Post-Leap Check 3.2 compared to simulations by Gillespie method 2.1 for various orders of the volumetric term V.

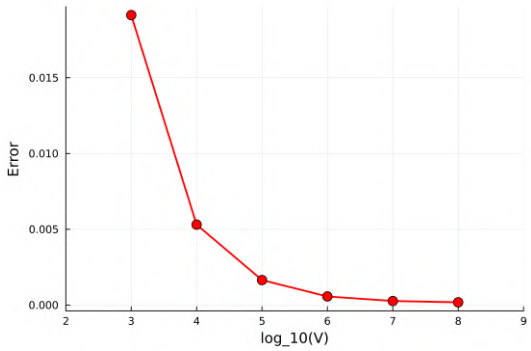


Figure 5.15: Expected errors of simulations of the model 5.1 by τ-leaping method with Post-Leap Check 3.2 compared to simulations by Gillespie method 2.1 for various orders of the volumetric term V.

The final approximation is valid due to the small value of the expected error.

5.5 Application of the Multinomial algorithm in Classical Scaling

Given the highly distinctive nature of model 5.1, a specific method has been devised to simulate it based on the multinomial distribution, hence referred to as the *Multinomial method*. The underlying concept of this algorithm is that a functional indicator of the maximum number of possible reactions is the current number of monomers: indeed, in each

state, there can be at most $\lfloor x_0/m \rfloor$ nucleations and at most x_0 growths. Thus, denoting as $p_0 = 1 - e^{-\lambda_0(x)\cdot\tau}$ the probability of there being at least one nucleation in the time window $(t, t+\tau)$, the number of nucleations is distributed as a binomial random variable with $n = \lfloor x_0/m \rfloor$ and $p = p_0$. The remaining monomers can either be utilized in nucleations or remain unused. Let $p_k = 1 - e^{-\lambda_k(x)\cdot\tau}$ be the probability of observing at least one growth of the nanoparticles of size k in the time window $(t, t+\tau)$. Let $N = \{N_i\}_i$ be the vector of the number of growths of particles of size i in the time window $(t, t+\tau)$, which will be distributed as a multinomial random vector with parameters $n = x_0(t) - N_0$, representing the number of remaining monomers, and $p = [p_1, \ldots, p_{V-1}, 1 - \sum_{k=1}^{V-1} p_k]$ where the last term indicates the probability for a monomer to be unused. Hence the algorithm:

Algorithm 5.1 Multinomial algorithm

Consider V monomers. Choose a fixed value for τ and initialize $j = 0$, $t_0 = 0$ and $x(t_0) = (V,0,\ldots,0)$.

Repeat the following steps while $x_0(t_j) > 0$:

1: Compute the nucleation rate $\lambda_0(x(t_j))$ and the growth rates $\lambda_1(x(t_j)), \ldots, \lambda_{V-1}(x_(t_j))$;

2: Generate N_0, the number of nucleations, as:

$$N_0 \sim \mathit{Binomial}\left(\lfloor \frac{x_0(t_j)}{m} \rfloor, 1 - e^{-\lambda_0\cdot\tau}\right);$$

3: Compute:

$$p = \left[1 - e^{-\lambda_1(x_j)\cdot\tau}, \ldots, 1 - e^{-\lambda_{V-1}(x_j)\cdot\tau}, 1 - \sum_{k=1}^{V-1} p_i\right]$$

4: Generate N, the number of growths, as:

$$N \sim \mathit{Multinomial}(x_0(t_j) - N_0, p);$$

5: $N_k \leftarrow \min\{N_k, x_k(t_j)\},\ k = 1, \ldots V-1$;

6: Update the time: $t_{j+1} = t_j + \tau$;

7: Update the state: $x_0(t_{j+1}) = x_0(t_j) - m\cdot N_0 - \sum_{k=1}^{V-1} N_k$, $x_m(t_{j+1}) = x_m(t_j) + N_0 - N_m$, $x_i(t_{j+1}) = x_i(t_j) - N_i + N_{i-1}$, $i = m+1, \ldots, V-1$, $x_V(t_{j+1}) = x_V(t_j) + N_{V-1}$;

8: $j \leftarrow j + 1$.

The only parameter to adjust is the time interval, denoted as τ: a strong dependence of performance on the time interval has been observed. Clearly, it is expected that as τ decreases, the computational cost increases while the error decreases. Therefore, for each V, a time interval was chosen that achieves a good trade-off between average computational time and strong approximation error, with particular attention given to finding a model that could compete with the Gillespie method.

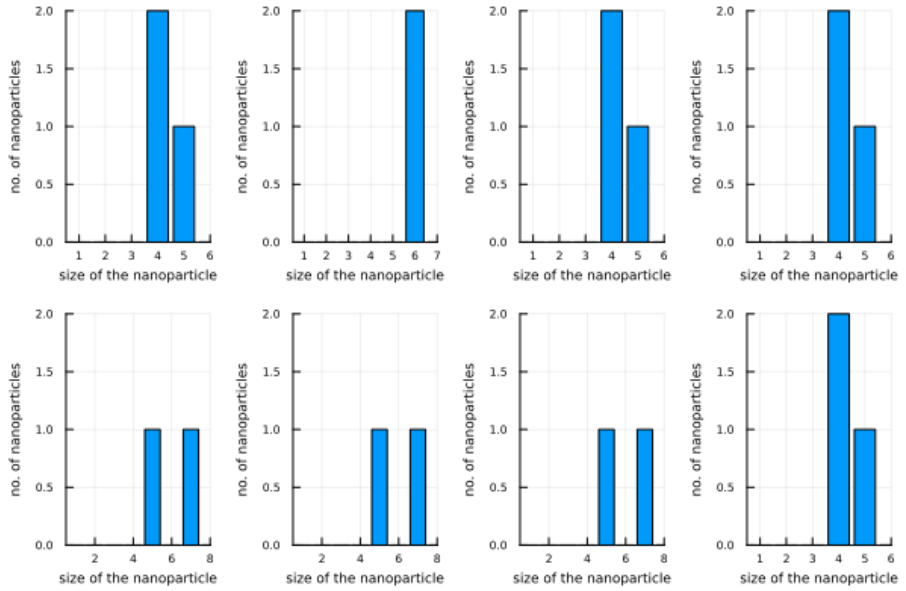


Figure 5.16: Simulations of the model 5.1 by Multinomial algorithm in Classical Scaling, $V = 10$, $\tau = 10^{-3}$.

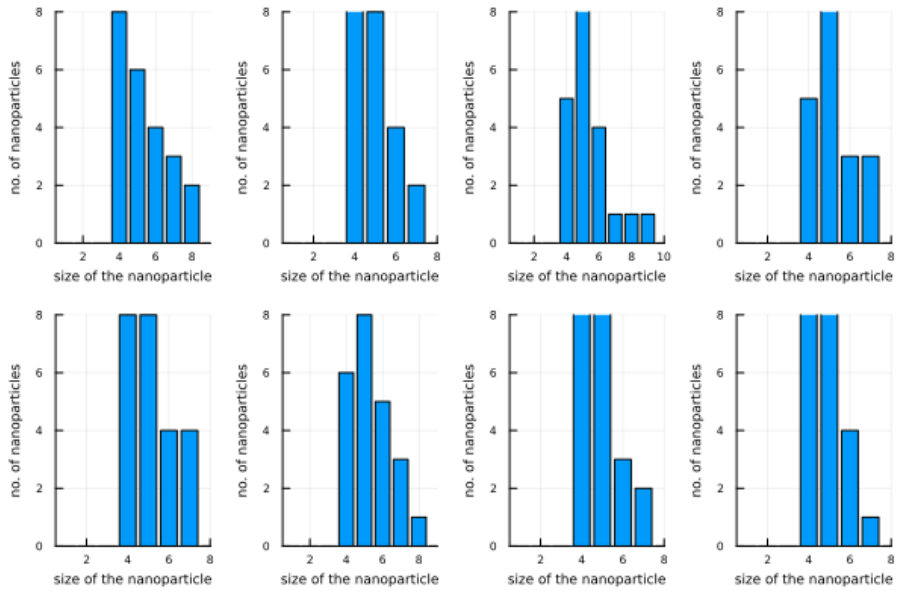


Figure 5.17: Simulations of the model 5.1 by Multinomial algorithm in Classical Scaling, $V = 10^2$, $\tau = 10^{-4}$.

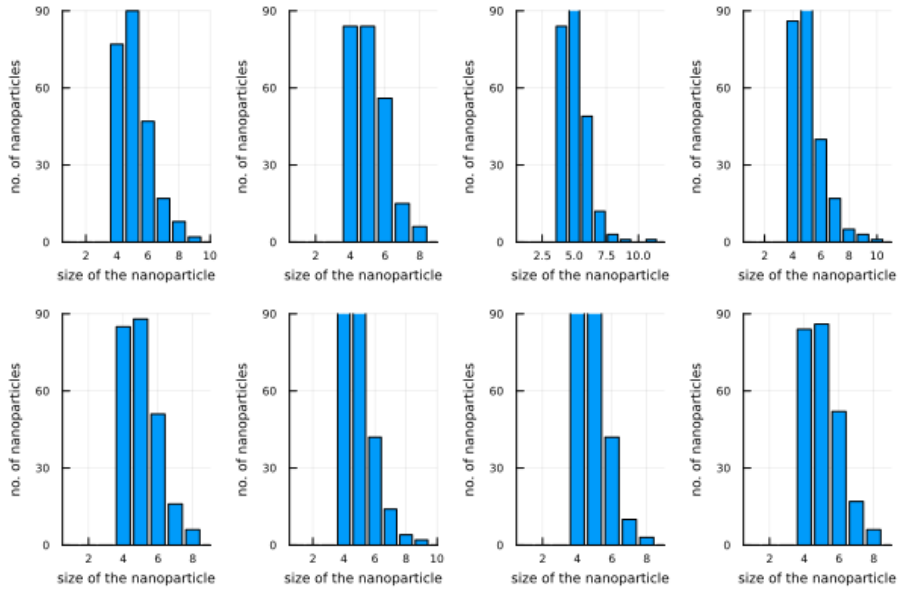


Figure 5.18: Simulations of the model 5.1 by Multinomial algorithm in Classical Scaling, $V = 10^3$, $\tau = 10^{-5}$.

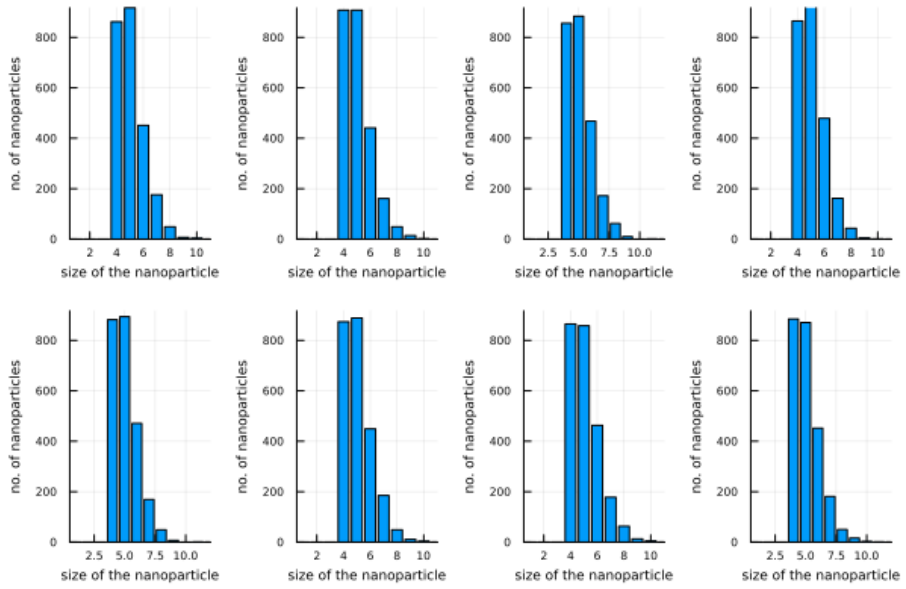


Figure 5.19: Simulations of the model 5.1 by Multinomial algorithm in Classical Scaling, $V = 10^4$, $\tau = 10^{-6}$.

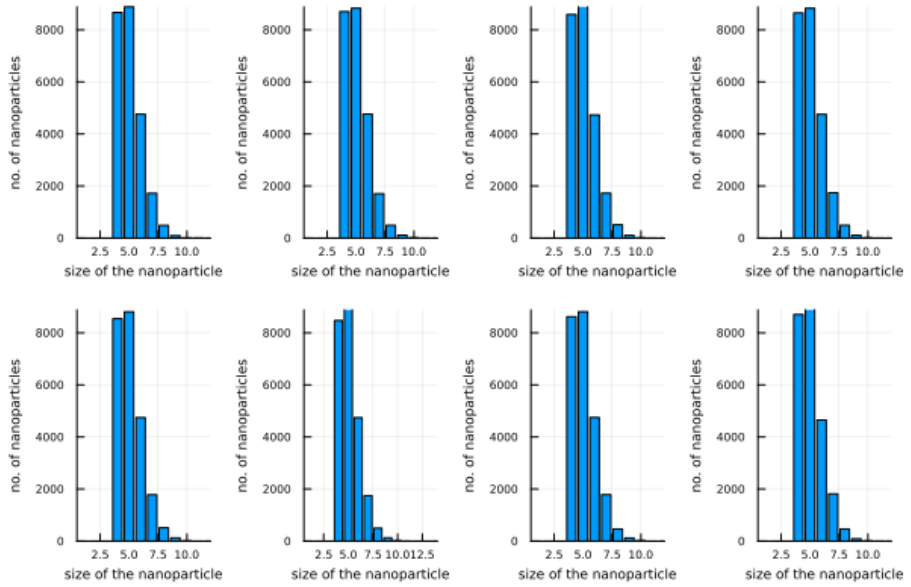
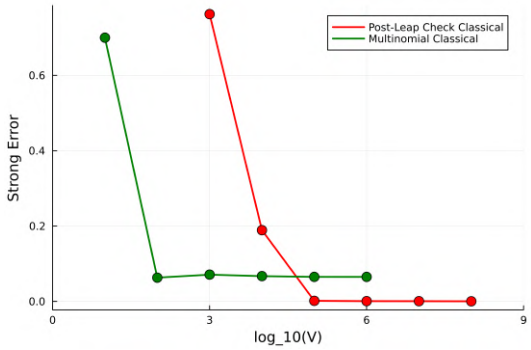


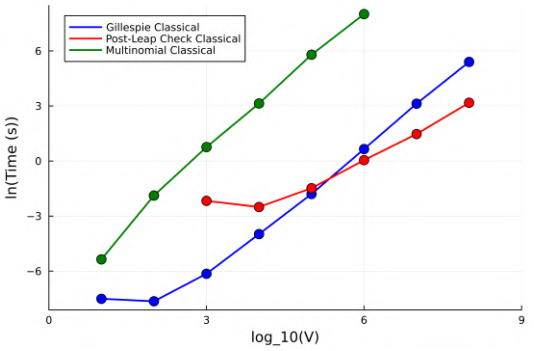
Figure 5.20: Simulations of the model 5.1 by Multinomial algorithm in Classical Scaling, $V = 10^5$, $\tau = 10^{-7}$.

Multinomial - Classical					
V	Time(s)	Error 1	Error 2	Error 3	Error 4
10	0.00471041	7.000000e-1	0.000000e0	7.603453e-2	1.082531e-1
10^2	0.15252837	4.495464e-2	2.797855e-2	4.464329e-2	7.817042e-2
10^3	2.13661898	1.281227e-2	1.328002e-2	3.330558e-2	7.099911e-2
10^4	23.0073423	1.120983e-2	1.052616e-2	2.943947e-2	6.698095e-2
10^5	327.856395	1.969799e-2	1.077690e-2	2.877598e-2	6.501643e-2
10^6	3179.58357	1.737343e-2	1.035374e-2	3.004737e-2	6.794372e-2
10^7	slow	slow	slow	slow	slow
10^8	slow	slow	slow	slow	slow

Table 5.4: Average computational times and strong approximation errors for $V = 10{,}10^2, \ldots, 10^8$ over 8 simulations of model 5.1 in Classical Scaling by Multinomial algorithm 5.1.



(a) Strong Error Analysis.



(b) Time Analysis.

Figure 5.21: Comparison of the strong approximation error and average computational times of the simulations of model 5.1 in Classical Scaling by the τ-leaping method with Post-Leap Check 3.2 and by Multinomial method 5.1 in relation to the order of the volumetric term V.

It is immediately noticeable how the multinomial method is slower than the Gillespie method for every analyzed V, thus offering no advantage. In fact, its use is also disadvantageous due to the introduction of error. It is interesting to note that despite the higher error compared to the τ-leaping method with Post-Leap Checks, this error tends to decrease as the volumetric term V increases.

5.6 Application of the Gillespie algorithm in Alternative Scaling

The results obtained so far are interesting from a computational perspective but not from the perspective of the simulation outcome itself. Much more interesting results are obtained in environments where the probabilities of nucleation and growth are of the same order already in the initial phase. Therefore, an alternative scaling is proposed:

$$\nu^V = \nu \cdot V^{1-m}; \quad \gamma^V = \gamma.$$

This allows for reaction rates, both for nucleation and growth, to be of the order of $\mathcal{O}(V)$. It is observed that in this scaling, the convergence result to the deterministic solution stated in 1.3 is no longer valid. However, this is not significant as the system of differential equations associated with the problem is potentially infinite, making it impossible to find a deterministic solution even computationally.

The alternative scaling is applied to the model, and simulations are conducted using the Gillespie algorithm. Below are the results for various values of the volumetric term V.

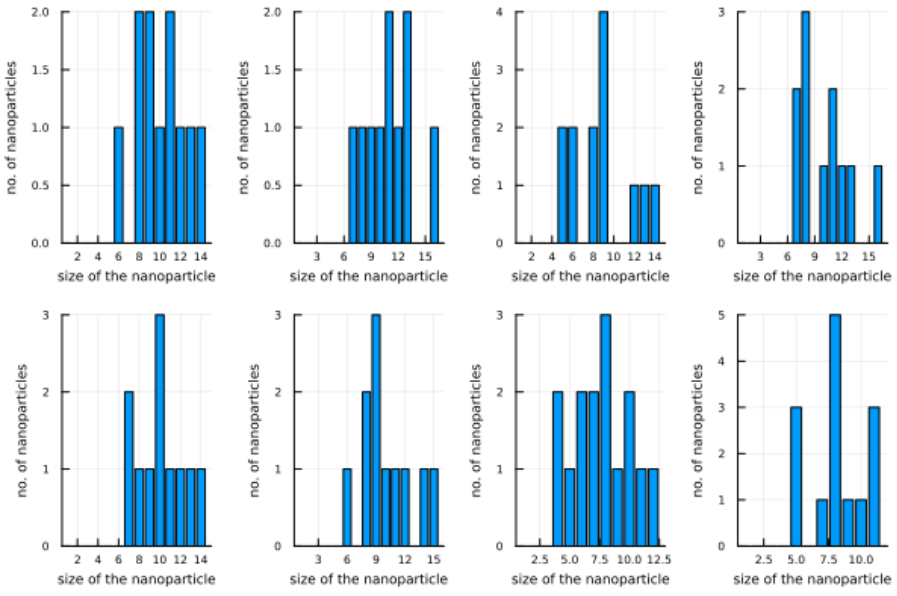


Figure 5.22: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Alternative Scaling, $V = 10^2$.

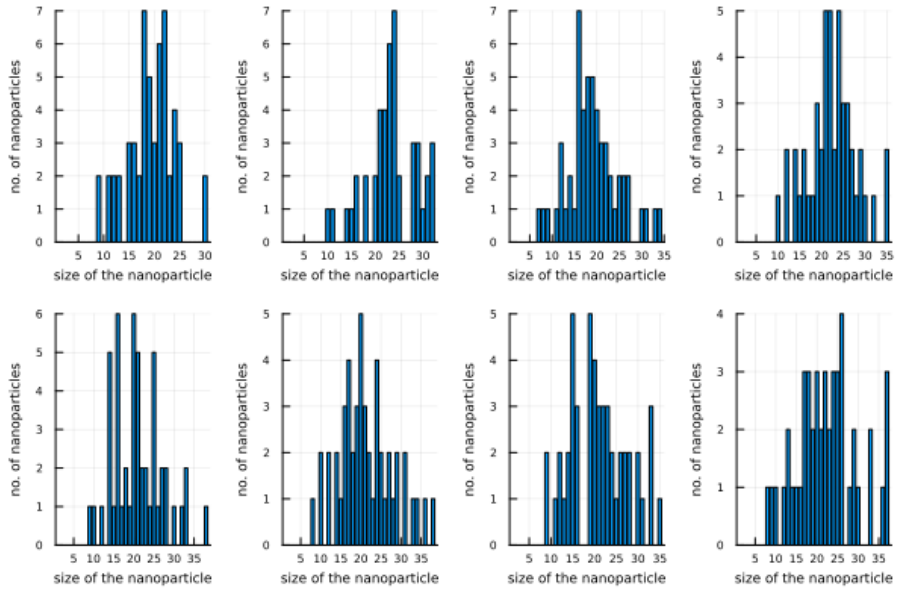


Figure 5.23: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Alternative Scaling, $V = 10^3$.

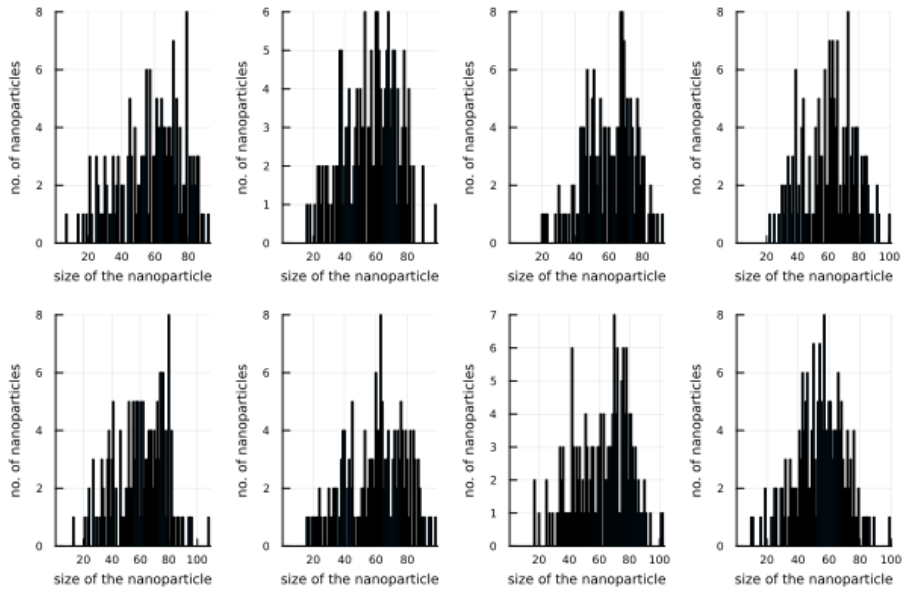


Figure 5.24: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Alternative Scaling, $V = 10^4$.

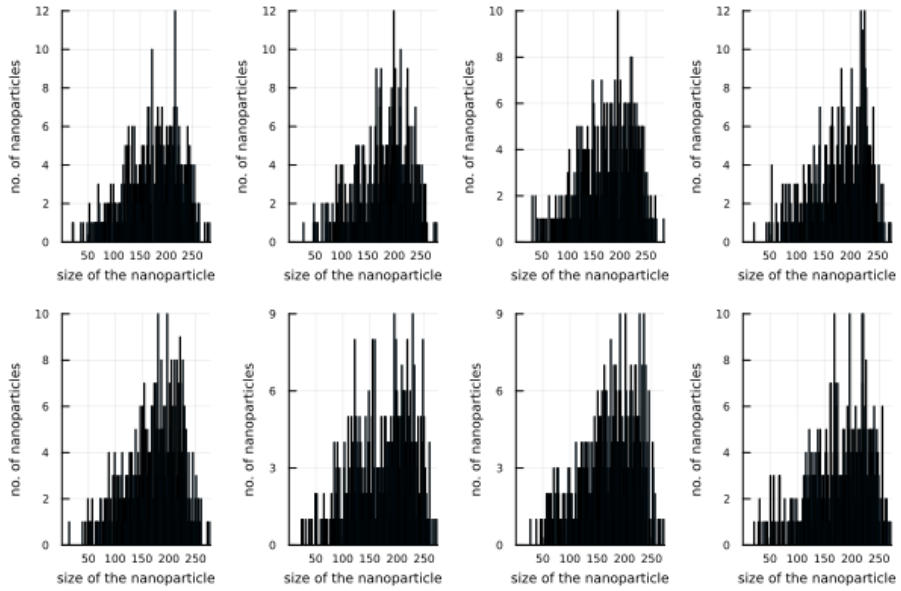


Figure 5.25: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Alternative Scaling, $V = 10^5$.

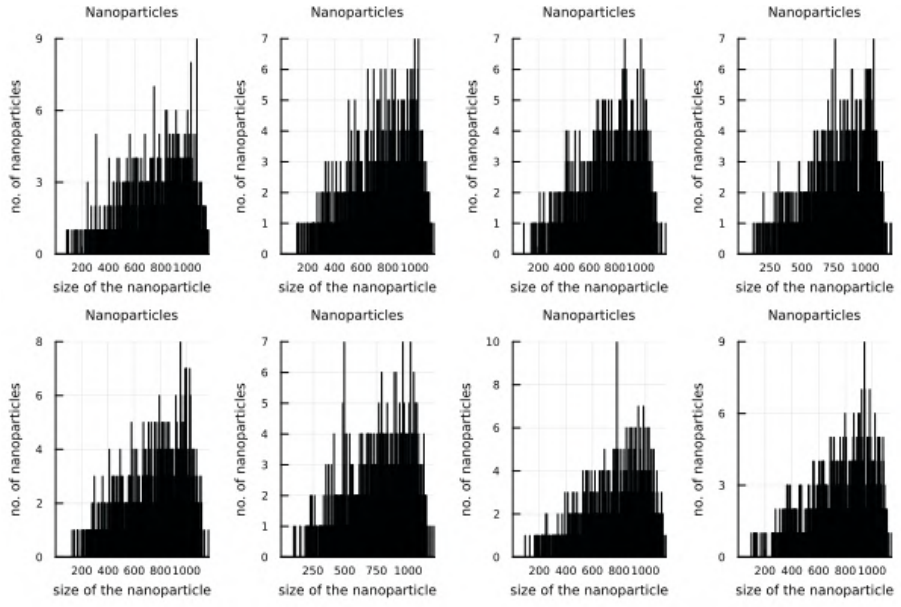


Figure 5.26: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Alternative Scaling, $V = 10^6$.

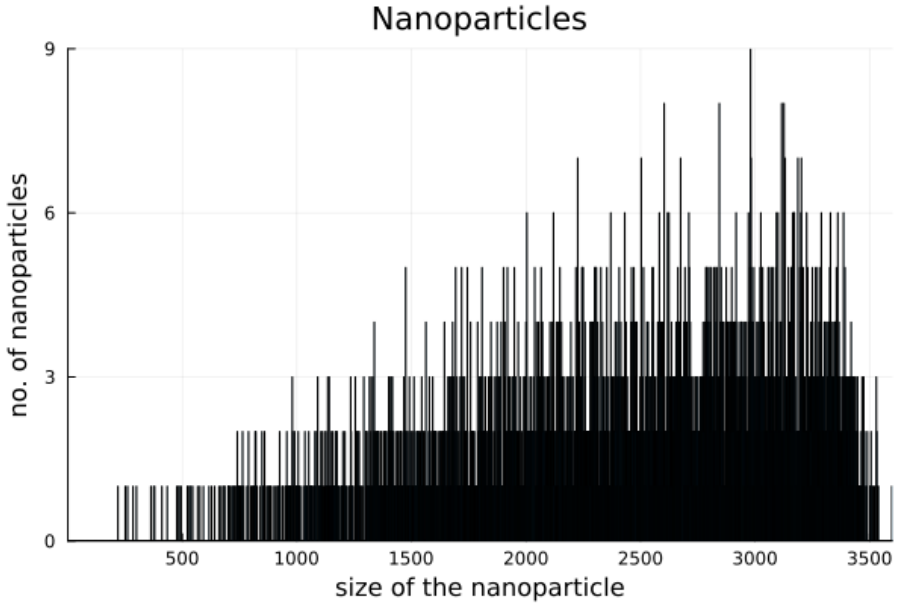


Figure 5.27: Simulations of the model 5.1 by Gillespie algorithm 2.1 in Alternative Scaling, $V = 10^7$.

In this case, the results are very different from what was observed in Classical Scaling. Since already in the early instants, the probability of observing nucleation and that of observing growth were of the same order, nanoparticles had more opportunities for growth compared to Classical Scaling. This led to a maximum created size that increases with V and is overall larger than 10, the maximum created size in Classical Scaling. Clearly, as the sum of sizes of each nanoparticle is constant with respect to V, increasing the maximum created size decreases the peak of the number of particles of the same size, which changes from being of the order of V to being around 10, regardless of the value of the volumetric term.

Gillespie	
V	Time (s)
10	0.000711507
10^2	0.001295547
10^3	0.010353837
10^4	0.250507800
10^5	6.193990864
10^6	184.9025709
10^7	6112.358574

Table 5.5: Average computational times over 10^2 simulations of the model 5.1 in Alternative Scaling by Gillespie algorithm 2.1 for various orders of the volumetric term V.

It is observed that in this case, the increase in computational times is much more

rapid with the increase in the volumetric term. This is primarily due to the fact that the maximum size created will be much larger than that for the Classical Scaling case: indeed this scaling allowed only for a reduction in memory storage to the order of $\mathcal{O}(V^{\frac{1}{2}})$.

5.7 Application of the τ-leaping algorithm with Post-Leap Check in Alternative Scaling

Given that this model is based on the choice of four parameters ε, p, p^*, q, a *Grid Search* algorithm has been applied, again and similarly to the classical scaling, to evaluate average computational times and strong approximation errors generated by the τ-leaping algorithm with Post-Leap Check for every possible combination of parameters (excluding combinations where $p^* < p$).The values were taken from $\varepsilon \in \{0.5\,0.6,\, 0.7,\, 0.8,\, 0.9\}$, $p \in \{0.1,\, \ldots,\, 0.9\}$, $p^* \in \{0.1,\, \ldots,\, 0.9\}$, $q \in \{0.1,\, \ldots,\, 0.9\}$. The following combinations were considered:

- $\varepsilon = 0.9;\ p = 0.4;\ p^* = 0.6;\ q = 0.9$ which produces the minimum strong approximation error, specifically with an error of 0.000329 and an average time of $32.504616250\,s$;

- $\varepsilon = 0.9;\ p = 0.3;\ p^* = 0.6;\ q = 0.9$ which produces the minimum average time, specifically with an error of 0.000329348 and an average time of $26.34556735\,s$.

Given the minimal difference in strong approximation error, the combination was chosen:

$$\varepsilon = 0.9; \quad p = 0.3; \quad p^* = 0.6; \quad q = 0.9$$

which results in much lower computational times. It resulted odd that almost all the combinations result in strong errors of the same order, while heavily differing in computational times.

Following the simulations for various values of the volumetric term V are reported.

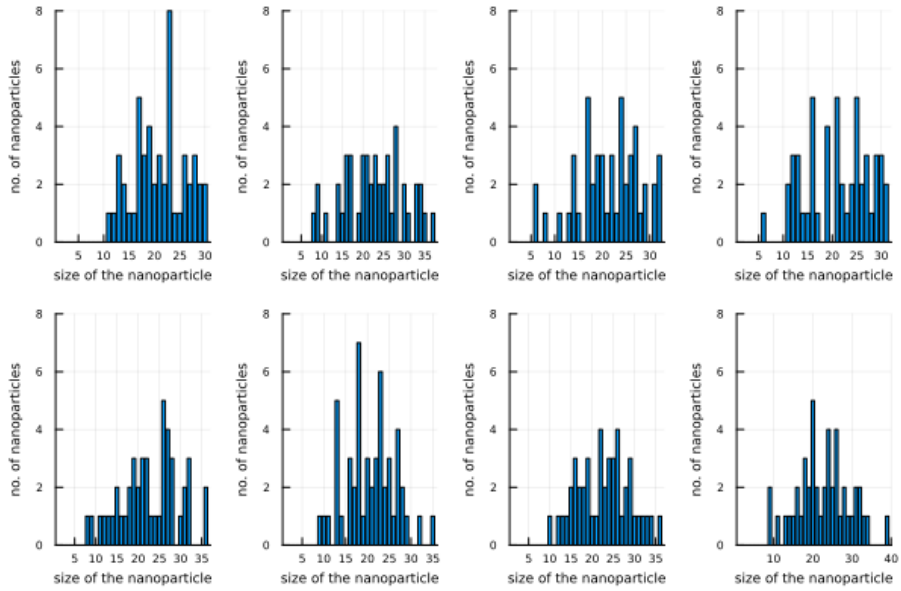


Figure 5.28: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Alternative Scaling, $V = 10^3$.

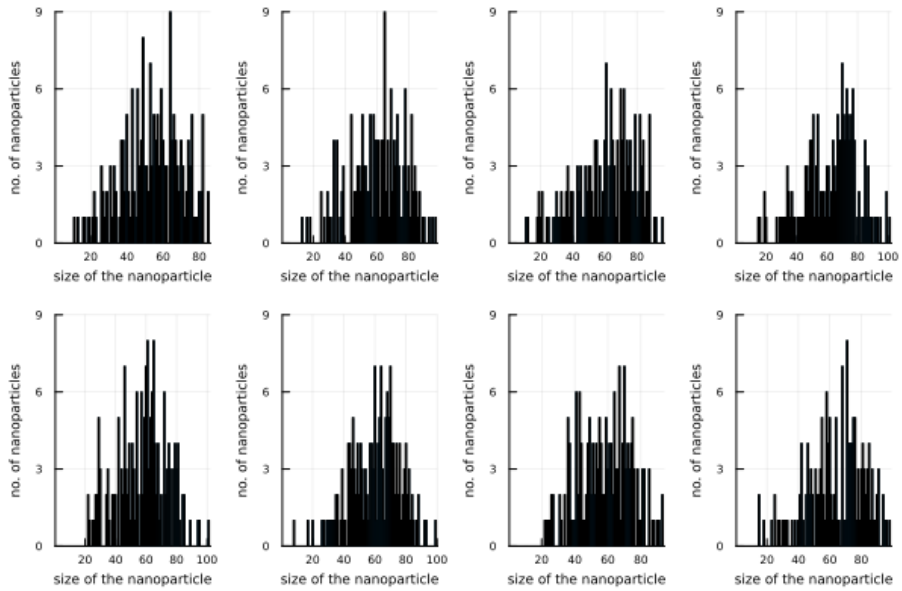


Figure 5.29: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Alternative Scaling, $V = 10^4$.

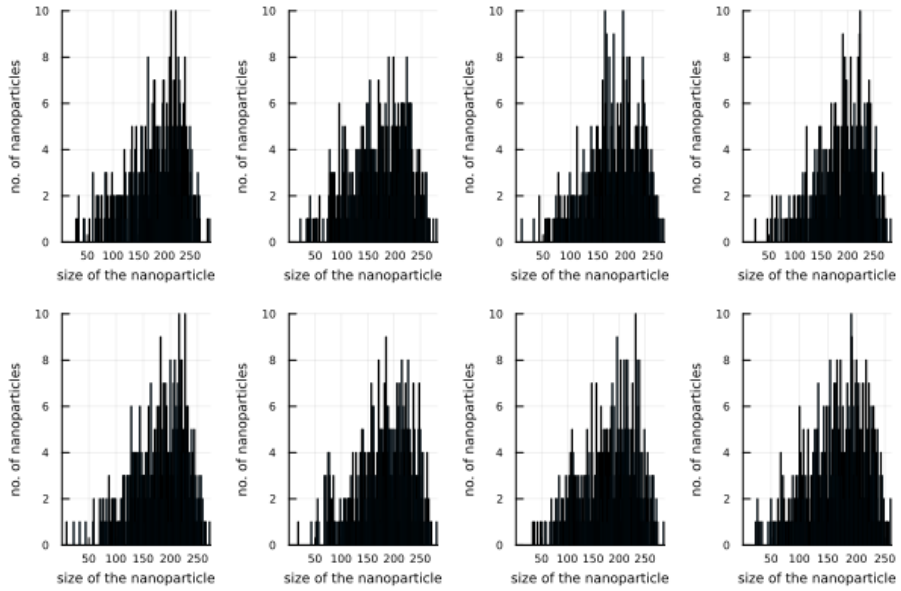


Figure 5.30: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Alternative Scaling, $V = 10^5$.

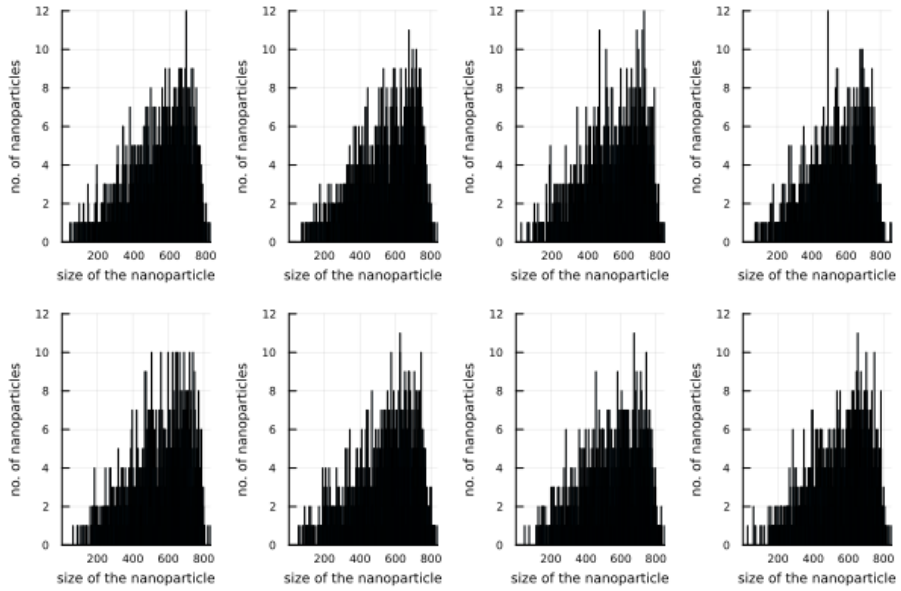


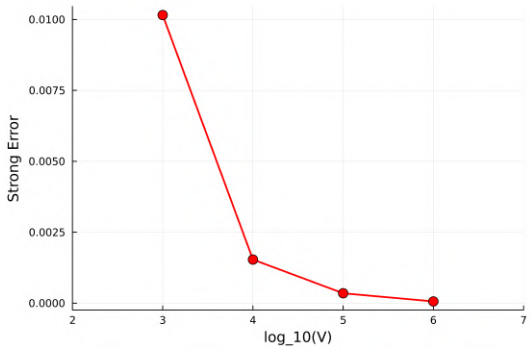
Figure 5.31: Simulations of the model 5.1 by τ-leaping algorithm with Post-Leap Check 3.2 in Alternative Scaling, $V = 10^6$.

To calculate the strong approximation error of the τ-leaping method with Post-Leap Check compared to the Gillespie algorithm, a study was conducted on the actual maximum

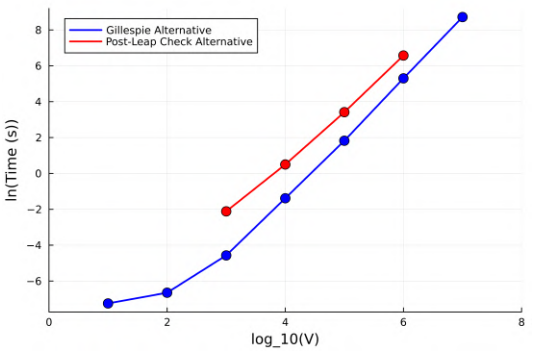
time at which the monomers are depleted. Four internal moments were chosen between 0.0 and this maximum time.

τ-leaping with Post-Leap Check - Alternative					
V	Time(s)	Error 1	Error 2	Error 3	Error 4
10	slow	slow	slow	slow	slow
10^2	slow	slow	slow	slow	slow
10^3	0.120402702	7.774963e-3	9.170888e-3	9.873224e-3	1.015499e-2
10^4	1.645742570	1.536668e-3	1.793654e-3	1.916876e-3	1.930803e-3
10^5	30.42989161	2.925356e-4	3.283135e-4	3.458503e-4	3.385290e-4
10^6	715.9502806	4.938355e-5	5.798645e-5	6.010340e-5	6.374605e-5
10^7	slow	slow	slow	slow	slow

Table 5.6: Average computational times and strong approximation errors for $V = 10{,}10^2, \ldots, 10^6$ over 10^2 simulations of model 5.1 in Alternative Scaling by τ-leaping with Post-Leap Check 3.2.



(a) Strong Error Analysis.



(b) Time Analysis.

Figure 5.32: Analysis of the strong approximation error and average computational times of the simulations of model 5.1 in Alternative Scaling by the τ-leaping method with Post-Leap Check 3.2 using $\varepsilon = 0.9$, $p = 0.3$, $p^* = 0.6$, $q = 0.9$ in relation to the order of the volumetric term V.

In the context of simulating model 5.1 under Alternative Scaling, it is evident from figure 5.32 that the τ-leaping method with Post-Leap Check consistently demonstrates inferior performance compared to the Gillespie method. This inferiority manifests as the approximated method has an actual non-zero strong approximation error while yielding longer mean computational times with respect to the exact method.

Furthermore, examination of plot 5.32 reveals a trend wherein the strong approximation error tends to diminish with escalating volumetric term V. This is proof to the fact that the simulations performed by Tau-Leap with Post-Leap Checks tends to the ones performed by Gillespie, but there is no proof of a convergence to a deterministic solution, as the hypotheses outlined in 1.3.2 are not satisfied in the new scaling.

Analogously, analysis of the expected error in the Euclidean norm of simulations conducted via the τ-leaping method with Post-Leap Check, relative to simulations conducted via the Gillespie method, yields insights into their comparative performance.

Expect errors	
V	Time (s)
10	slow
10^2	slow
10^3	1.011814e-2
10^4	1.887261e-3
10^5	3.416598e-4
10^6	6.059664e-5
10^7	slow

Table 5.7: Expected errors of simulations of the model 5.1 by τ-leaping method with Post-Leap Check 3.2 compared to simulations by Gillespie method 2.1 for various orders of the volumetric term V.

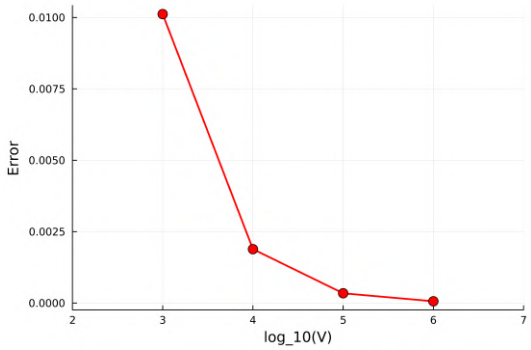


Figure 5.33: Expected errors of simulations of the model 5.1 by τ-leaping method with Post-Leap Check 3.2 compared to simulations by Gillespie method 2.1 for various orders of the volumetric term V.

The final approximation is valid due to the small value of the expected error.

5.8 Application of the Multinomial algorithm in Alternative Scaling

The multinomial method is used to simulate model 5.1. The results for various values of the volumetric term V are presented below.

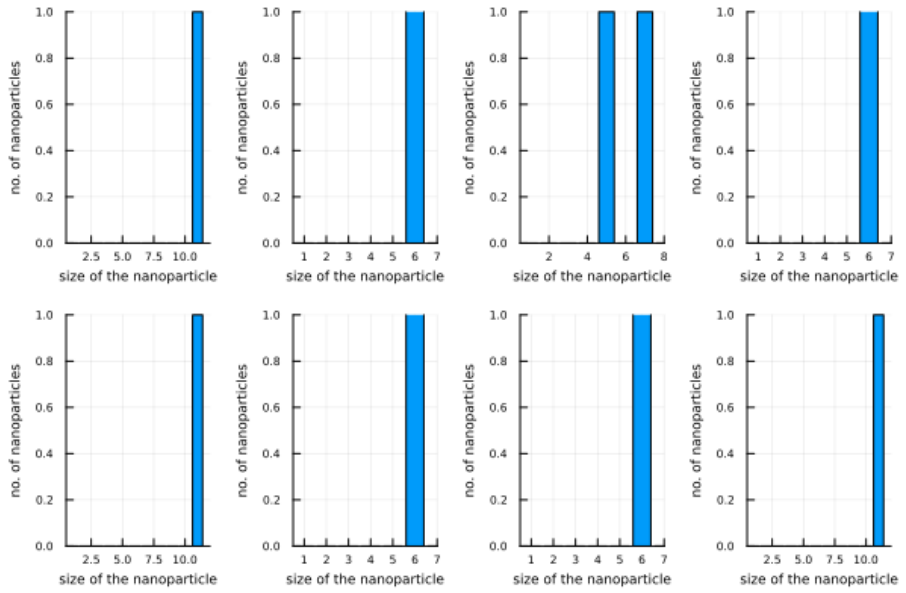


Figure 5.34: Simulations of the model 5.1 by Multinomial algorithm in Alternative Scaling, $V = 10^1$, $\tau = 10^{-3}$.

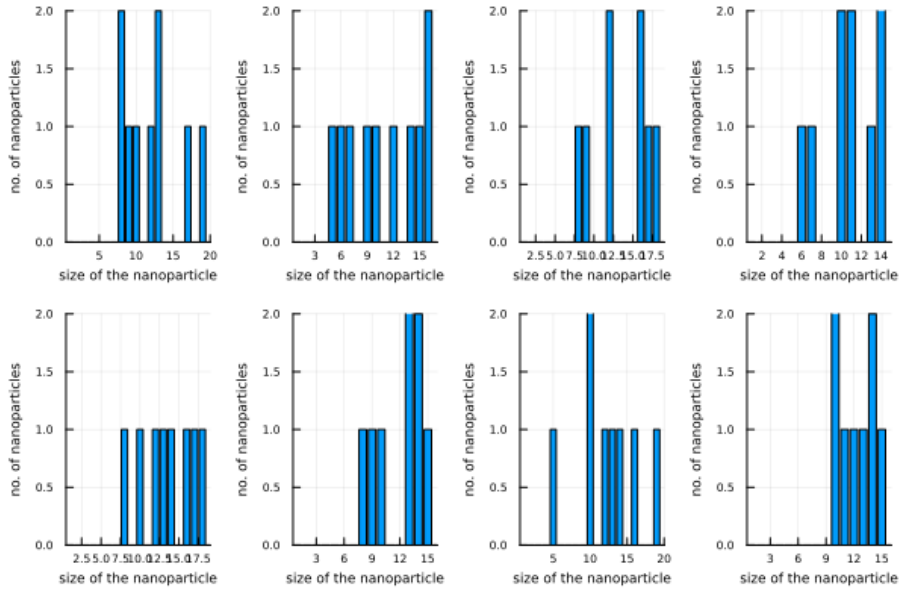


Figure 5.35: Simulations of the model 5.1 by Multinomial algorithm in Alternative Scaling, $V = 10^2$, $\tau = 10^{-4}$.

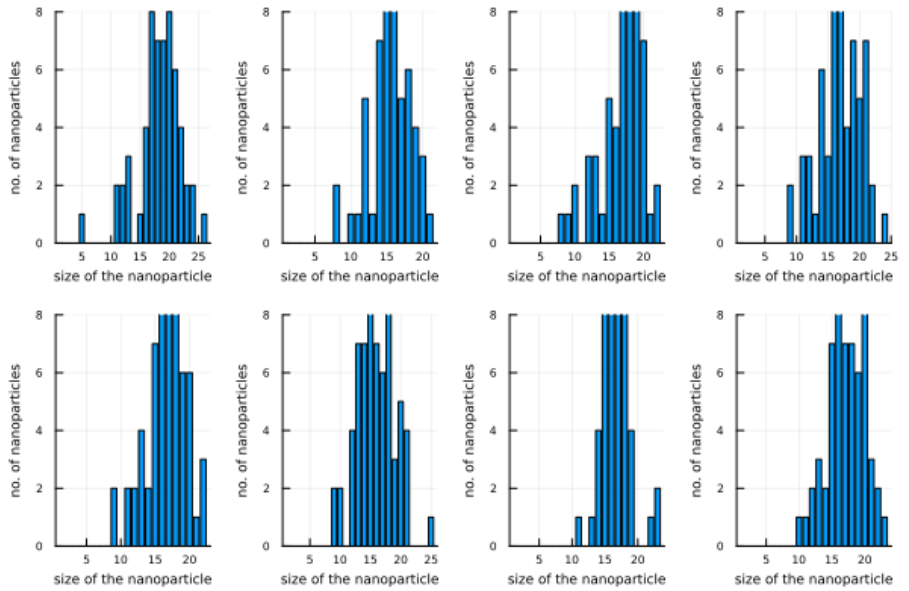


Figure 5.36: Simulations of the model 5.1 by Multinomial algorithm in Alternative Scaling, $V = 10^3$, $\tau = 10^{-5}$.

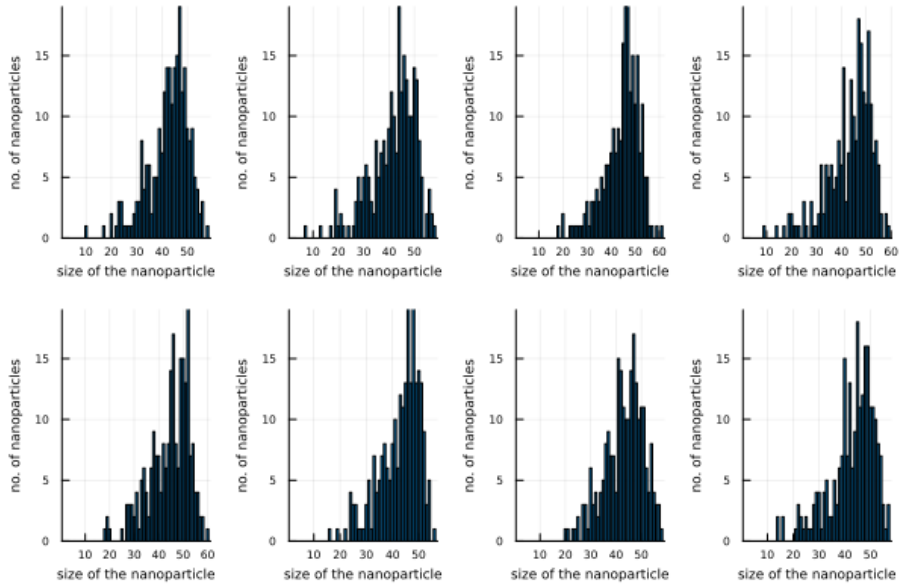


Figure 5.37: Simulations of the model 5.1 by Multinomial algorithm in Alternative Scaling, $V = 10^4$, $\tau = 10^{-7}$.

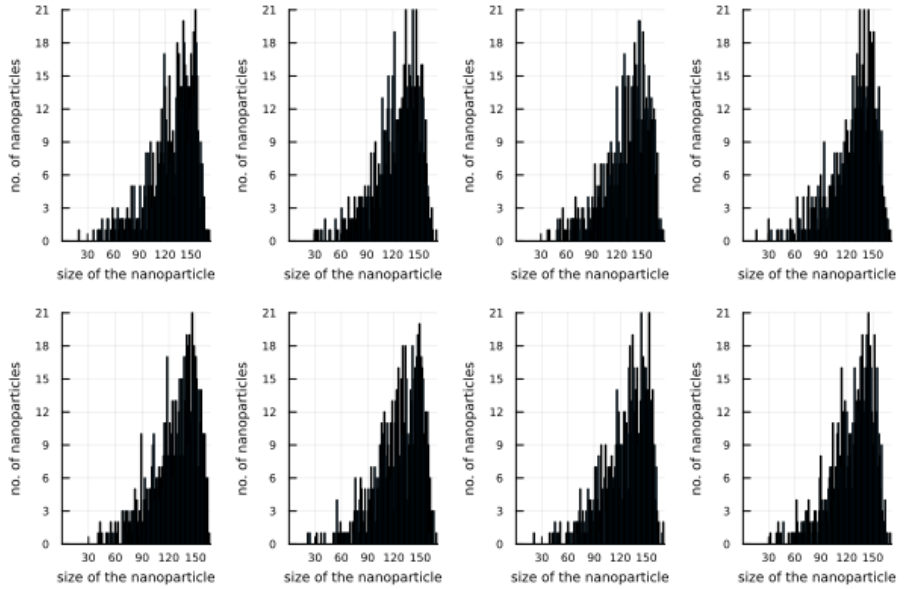


Figure 5.38: Simulations of the model 5.1 by Multinomial algorithm in Alternative Scaling, $V = 10^5$, $\tau = 10^{-9}$.

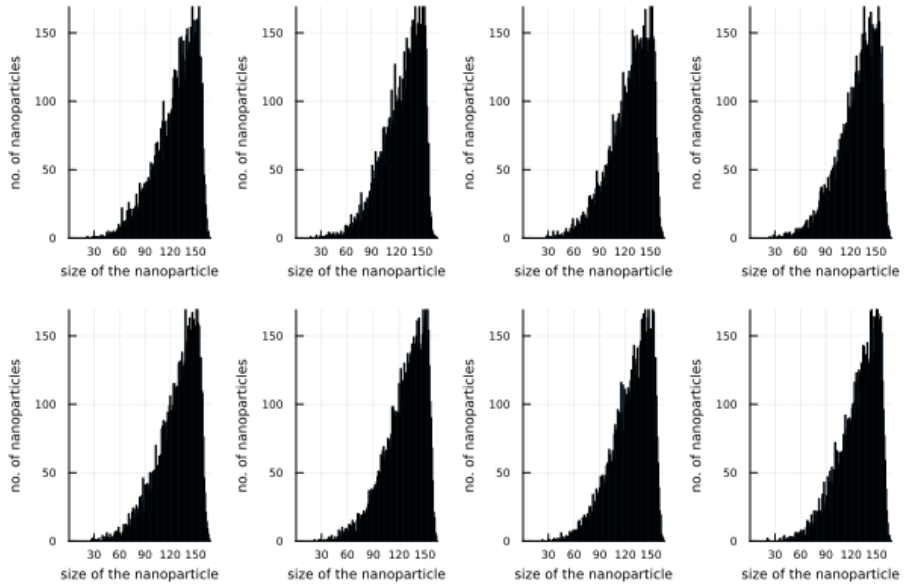
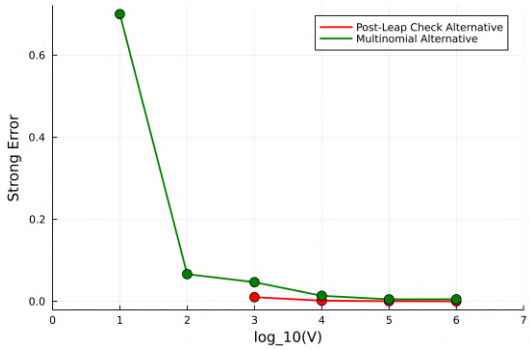


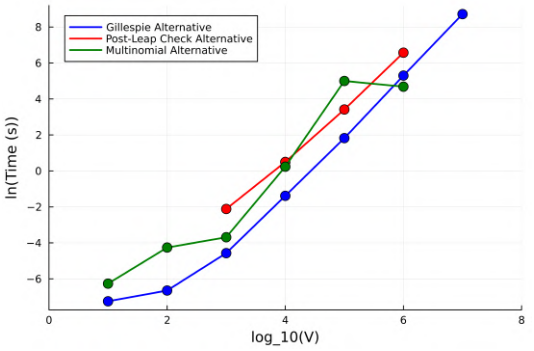
Figure 5.39: Simulations of the model 5.1 by Multinomial algorithm in Alternative Scaling, $V = 10^6$, $\tau = 10^{-10}$.

Multinomial - Alternative					
V	Time(s)	Error 1	Error 2	Error 3	Error 4
10	0.00190420	7.000000e-1	0.000000e0	2.591081e-1	1.276369e-1
10^2	0.01410978	5.922169e-2	5.131680e-2	5.850676e-2	5.179846e-2
10^3	0.02502164	4.661666e-2	4.033766e-2	4.513285e-2	2.887287e-2
10^4	1.26597972	1.125122e-2	1.350050e-2	1.253056e-2	7.961538e-3
10^5	148.544128	3.223722e-3	4.681665e-3	4.685433e-3	1.803056e-3
10^6	108.264683	3.278800e-3	4.111521e-3	5.090381e-3	5.112922e-3
10^7	slow	slow	slow	slow	slow

Table 5.8: Average computational times and strong approximation errors for $V = 10{,}10^2, \ldots, 10^7$ over 8 simulations of model 5.1 in Alternative Scaling by Multinomial algorithm.



(a) Strong Error Analysis.



(b) Time Analysis.

Figure 5.40: Comparison of the strong approximation error and average computational times of the simulations of model 5.1 in Alternative Scaling by the τ-leaping method with Post-Leap Check 3.2 and by Multinomial method 5.1 in relation to the order of the volumetric term V.

The computational costs of the multinomial method are similar to those of the τ-leaping method with Post-Leap Checks. However, it exhibits a higher level of strong approximation error. Nevertheless, for $V > 10^6$, the computational time of the multinomial method is observed to be lower than that of the Gillespie method. Moreover, the error tends to stabilize at low levels, rendering the multinomial method an effective simulation approach for scenarios with high cardinalities.

Bibliography

[1] P. Waage and C. Guldberg, "Studier over affiniteten forhandlinger," *VIdenskabs-Selskabet i ChrIstIana*, vol. 35, pp. 111–120, 01 1864.

[2] C. M. Guldberg and P. Waage, "Ueber die chemische affinität. § 1. einleitung," *Journal für Praktische Chemie*, vol. 19, no. 1, pp. 69–114, 1879.

[3] F. J. M. Horn and R. Jackson, "General mass action kinetics," *Archive for Rational Mechanics and Analysis*, vol. 47, pp. 81–116, 1972.

[4] M. Feinberg, "Complex balancing in general kinetic systems," *Archive for Rational Mechanics and Analysis*, vol. 49, pp. 187–194, 01 1972.

[5] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[6] T. G. Kurtz, "Representations of markov processes as multiparameter time changes," *The Annals of Probability*, vol. 8, no. 4, pp. 682–715, 1980.

[7] D. Anderson and T. Kurtz, *Stochastic Analysis of Biochemical Systems*. Springer International Publishing, 2015.

[8] S. N. Ethier and T. G. Kurtz, *Markov processes – characterization and convergence*. Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics, John Wiley & Sons Inc., 1986.

[9] D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *The Journal of Chemical Physics*, vol. 115, pp. 1716–1733, 07 2001.

[10] D. T. Gillespie and L. R. Petzold, "Improved leap-size selection for accelerated stochastic simulation," *The Journal of Chemical Physics*, vol. 119, pp. 8229–8234, 10 2003.

[11] Y. Cao, D. T. Gillespie, and L. R. Petzold, "Efficient step size selection for the tau-leaping simulation method," *The Journal of Chemical Physics*, vol. 124, p. 044109, 01 2006.

[12] A. J. Lotka, "Analytical note on certain rhythmic relations in organic systems," *Proceedings of the National Academy of Sciences*, vol. 6, no. 7, pp. 410–415, 1920.

[13] V. Volterra, *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*. Memorie della Classe di Scienze Fisiche, Matematiche e Naturali, Societá anonima tipografica "Leonardo da Vinci", 1927.

[14] A. Chatterjee, D. G. Vlachos, and M. A. Katsoulakis, "Binomial distribution based tau-leap accelerated stochastic simulation," *The Journal of Chemical Physics*, vol. 122, p. 024112, 12 2004.

[15] Y. Cao, D. T. Gillespie, and L. R. Petzold, "Avoiding negative populations in explicit Poisson tau-leaping," *The Journal of Chemical Physics*, vol. 123, p. 054104, 08 2005.

[16] T. Tian and K. Burrage, "Binomial leap methods for simulating stochastic chemical kinetics," *The Journal of Chemical Physics*, vol. 121, pp. 10356–10364, 11 2004.

[17] D. F. Anderson, "Incorporating postleap checks in tau-leaping," *The Journal of Chemical Physics*, vol. 128, Feb. 2008.

[18] M. Rathinam, L. Petzold, Y. Cao, and D. Gillespie, "Consistency and stability of tau-leaping schemes for chemical reaction systems," *Society for Industrial and Applied Mathematics*, vol. 4, pp. 867–895, 01 2005.

[19] T. Li, "Analysis of explicit tau-leaping schemes for simulating chemically reacting systems," *Multiscale Modeling & Simulation*, vol. 6, no. 2, pp. 417–436, 2007.

[20] D. F. Anderson, A. Ganguly, and T. G. Kurtz, "Error analysis of tau-leap simulation methods," *The Annals of Applied Probability*, vol. 21, Dec. 2011.

[21] D. Anderson, G. Enciso, and M. Johnston, "Stochastic analysis of biochemical reaction networks with absolute concentration robustness," *Journal of the Royal Society, Interface / the Royal Society*, vol. 11, p. 20130943, 04 2014.

[22] V. Berry and R. F. Saraf, "Self-assembly of nanoparticles on live bacterium: An avenue to fabricate electronic devices," *Angewandte Chemie International Edition*, vol. 44, no. 41, pp. 6668–6673, 2005.

[23] C. Tsai, A. Shiau, S. Chen, Y. Chen, P. Cheng, M. Chang, D. Chen, C. Chou, C. Wang, and C. Wu, "Amelioration of collagen-induced arthritis in rats by nanogold," *Arthritis & Rheumatism*, vol. 56, no. 2, pp. 544–554, 2007.

[24] J. N. Swanson, "Repeated Colloidal Gold Tests in Rheumatoid Arthritis," *Ann Rheum Dis*, vol. 8, pp. 232–237, Sep 1949.

[25] G. T. Jaeger, S. Larsen, N. li, and L. Moe, "Two years follow-up study of the pain-relieving effect of gold bead implantation in dogs with hip-joint arthritis," *Acta Vet Scand*, vol. 49, p. 9, Mar 2007.

[26] N. authors listed, "Gold is newest weapon in battle against Alzheimer's," *Health News*, vol. 12, p. 10, Mar 2006.

[27] J. D. Gibson, B. P. Khanal, and E. R. Zubarev, "Paclitaxel-functionalized gold nanoparticles," *J Am Chem Soc*, vol. 129, pp. 11653–11661, Sep 2007.

[28] X. Qian, X. H. Peng, D. O. Ansari, Q. Yin-Goen, G. Z. Chen, D. M. Shin, L. Yang, A. N. Young, M. D. Wang, and S. Nie, "In vivo tumor targeting and spectroscopic detection with surface-enhanced Raman nanoparticle tags," *Nat Biotechnol*, vol. 26, pp. 83–90, Jan 2008.

[29] V. K. LaMer and R. H. Dinegar, "Theory, production and mechanism of formation of monodispersed hydrosols," *Journal of the American Chemical Society*, vol. 72, no. 11, pp. 4847–4854, 1950.

[30] J. Turkevich, P. C. Stevenson, and J. Hillier, "A study of the nucleation and growth processes in the synthesis of colloidal gold," *Discuss. Faraday Soc.*, vol. 11, pp. 55–75, 1951.

List of Figures

List of Tables

Lists of algorithms

Acknowledgements

É doveroso e al contempo un piacere per me ringraziare le persone che hanno contribuito a questo percorso.

Ringrazio il prof. Enrico Bibbona per la sua esperta guida nella stesura di questo lavoro e per il supporto durante il processo di studio.