ADAPTIVE STEP-SIZE SELECTION FOR TAU-LEAPING SIMULATION OF STOCHASTIC PARTICLE COALESCENCE

BY

CHOWDHURY MAHBUB ASHRAF

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Adviser:

Assistant Professor Matthew West
ABSTRACT

This study develops a group of new leap selecting procedures and acceptance criteria for use with the tau leaping method for stochastically simulating the particle coalescence problem. Existing work provides leap selection procedures for pre-leap methods and leap acceptance criteria for post-leap methods. These existing methods have been primarily designed for stochastic chemical reaction problems and are not necessarily well-suited for the increased number of possible events present in coagulation and coalescence simulations. In this work, simple and computationally efficient conditions are developed that perform well for coagulation problems, both for pre-leap selection and post-leap acceptance testing for tau leaping simulation. Theoretical background for these conditions is provided and their performance is investigated numerically. A comparative study is provided of error and computational cost using the pre-existing and newly developed conditions for both pre-leap and post-leap tau selection.
To my parents, for their love and support. To my adviser for guiding me throughout. To my wife to be with me always.
ACKNOWLEDGMENTS

This work has been completed with the support of so many people. Without them it would not be possible. Many thanks to my adviser Professor Matthew West who guided me throughout my Master degree. His inspirational guidance was my motivation. So many times I came up with stupid questions, and he answered them with utmost patience. It was a great pleasure to work with him. I would also like to thank Prof. Nicole Reimer and Prof. Lee Deville for their comments and suggestions at different stages of my work. I was blessed with group members Rishabh Jain, Pradeep Gudla and Peter Maginnis who were always beside me, and helped to in so many occasions. I would also like to thank other members of PartMC group for providing a nice friendly atmosphere.

Thanks to Almighty for blessing me with the nicest parents in the universe. It is my parents who are and will be my inspiration throughout my life. I am truly grateful to my parents. Love you Abbu and Mamoni.

Numerous friends have kept me motivated during my period of study. I want to express my sincere gratitude towards them.

Finally, special thanks to my wife Afsana Bahar Trini for supporting me whenever I needed that most.

Nevertheless, thanks to almighty Allah.
# TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION ............................................. 1

CHAPTER 2 SMOLUCHOWSKI’S COALESCENCE EQUATION
   AND STOCHASTIC SIMULATION ALGORITHM ..................... 7
   2.1 Smoluchowski’s Coalescence Equation ......................... 7
   2.2 Analytical Solution ........................................... 9
   2.3 Solution from ODE solvers ..................................... 9
   2.4 The Stochastic Simulation Algorithm (SSA) ................... 10
   2.5 Monte Carlo Step ............................................. 13
   2.6 Comparison between Solutions ................................ 15

CHAPTER 3 ADAPTIVE TAU LEAPING METHODS ..................... 21
   3.1 Basic Idea ................................................... 21
   3.2 Constant Tau Leaping ......................................... 22
   3.3 Stochastic Chemical Kinetics .................................. 23
   3.4 Pre Leap Method .............................................. 25
   3.5 Post Leap Method ............................................. 32
   3.6 New Approach of Selecting Leap Conditions ................. 39
   3.7 Fixing Different Leap Conditions .............................. 42

CHAPTER 4 NUMERICAL RESULTS ................................. 47
   4.1 Kernels and Initial Conditions ............................... 47
   4.2 Time Steps Chosen by Different Tau Leaping Methods .... 48
   4.3 Error Vs Cost Plots .......................................... 50
   4.4 Selecting the Controlling Parameter $\varepsilon$ in Pre Leap ... 55

CHAPTER 5 CONCLUSION ............................................. 58

REFERENCES ...................................................... 59
A stochastic process, which is the counterpart of deterministic process, tries to capture any type of randomness present in the system. It is a statistical process where a number of random parameters evolve with time or space. These randomnesses might be significant, and we need to have an idea about their fluctuations. In deterministic process, we will obtain the same result every time we repeat for a given initial condition. However, stochastic process deals with every possible solutions by considering all the randomnesses present in the system given by some probability distribution. Thus, even if we try to solve the same problem with same initial condition multiple times, we will come up with different solution paths showing how the process might develop with time. After repeating the procedure multiple times, the average of all the solutions might give us an idea about what should we expect at a certain point of time.

Now a days, the stochastic process has caught the attention of many scientists and researchers, as there are a lot of processes with randomness, and that random behavior is the main concern to them. Also sometimes many deterministic systems are way too complex to solve them analytically or numerically using the existing methods. In those cases, analytical solutions might not available, and it is too expensive to come up with a numerical solution. Thus, scientists and engineers are now using stochastic models in different varieties of problems. Braumann et al. [6] have applied stochastic model for granular material simulation, Riemer et al. [25] have used it in aerosol particle simulation. Also Griesemer et al. [19] and Samad et al. [26] used stochastic models for biological systems and Gillespie [15] used this for coupled chemical reacions. These types of models are also extensively used in the field of finance and economics; such as in risk management by Lybbert et al. [20].
The main disadvantage of using stochastic model is, we have to run the simulation multiple times to get the expected picture, and we need to deal with a large number of particles to ensure better convergence. This is often a great drawback because of higher computer cost associated with the total process. Thus, in recent years, developing time efficient and cost effective algorithms for different stochastic processes has become one of the major concerns of people in this field.

In recent years, statistical models for particle growth have been addressed by many researches because of its considerable application in various fields of science like physics, chemistry and meteorology. One of the major applications of this statistical model is the growth of cloud droplets in raindrop formation. Warm rain formation in clouds is initiated by collision and coalescence by cloud droplets. Given an initial distribution of cloud droplets, the rate at which the coagulation will take place is determined by the mass of the droplets presented in the cloud along with the coalescence kernel, and also the initial number of particles in the system. The coalescence process leads to a shift of the initial distribution which contains larger cloud droplets as the particle size grows due to coagulation. This ultimately results in the formation of rain drops. Those particles which have collected sufficient mass through this coalescence process fall out of a cloud.

Chemical reactions between different chemical species is another example of this type of problems, where different particles react with each other depending on the rate of reaction.

The deterministic model for particle growth was first proposed by Smoluchowski [27]. A lot of work has been done to solve the proposed ODE both analytically and numerically. The governing differential equation was termed as Smoluchowski’s Coalescence Equation (SCE). However, it was found that, the analytical solution of SCE is only available for a few simple kernels. In his paper at 1972, Gillespie [13] developed the stochastic model for the same problem. To do that, he had to make some assumptions, and ignore some correlations. Since then, numerous work have been performed to develop a faster and efficient algorithm to solve these types of models.
In 1975, Gillespie [14] proposed an algorithm to solve the stochastic model of coalescence problem. In developing the algorithm, he took care of all the assumptions and correlations which he assumed in developing the model. The algorithm was known as Stochastic Simulation Algorithm (SSA), and it is an exact algorithm because it simulates only one event at a time. Later, he extended this algorithm for coupled chemical reaction system [15, 16].

Since then, Gillespie’s algorithm has been used in different applications, and it has become the backbone of stochastic simulation of coagulation process. A lot of work have been done in various aspects of this proposed algorithm to improve the basic one. Gillespie ignored the gelation phenomenon which is the loss of mass during coagulation in his algorithm. This problem was addressed by Eibeck and Wagner [11], where they proposed a new algorithm which will take care of this fact. With a new acceptance rejection technique, this algorithm worked much efficiently if the kernel is chosen carefully. Then Eibeck and Wagner [12] proposed another new algorithm with reduced variance in order to lower both the systemic and statistical cost in simulation and ensure lower statistical fluctuation than direct simulation.

Vikhansky and Kraft [30] proposed the single particle method which is a Monte Carlo algorithm based on single particle. It was a simple modification of Gillespie’s algorithm. This iterative method does not depend on the state of the any of the neighboring particles. Vikhansky and Kraft [29] also performed an identification and sensitivity analysis of different algorithms. Recently Man et al. [21] proposed coupling algorithms to calculate the parametric derivatives of coagulation equation using central difference estimator. Bailleul et al. [4] also developed algorithms for the same purpose using stochastic Monte Carlo method. In both the cases, emphasis was given to variance reduction by reducing the difference between the sample trajectories.

A major concern about Gillespie’s algorithm was the cost. It simulates only one event at a time and if a huge number of particle is present in the system, then it takes quite a long time to simulate the whole process. A great amount of work have been performed by Daniel Gillespie and other researchers to get rid of this major drawback of the algorithm. To accelerate the simulation
speed, Gillespie [17] proposed the tau leaping algorithm. It was originally proposed for chemical reaction system. In this, a leap along the time axis is made, and the number of each possible event during the leap is calculated. Thus, one can obtain better simulation speed by sacrificing some of its accuracy.

In the tau leaping algorithm, we have to choose a time step $\tau$ at the beginning of the leap, provided that a specific leap condition is not violated. This leap condition ensures the accuracy of our simulation with higher speed. The main challenge of using tau leaping algorithm was to find out a viable leap condition for the process and then develop an algorithm to generate $\tau$ that satisfies the leap condition. Cao et al. [8], Gillespie [17], Gillespie and Petzold [18] all tried different approaches to solve this problem for stochastic chemical reaction process, but until now there is no universal rule that satisfies the leap condition for different processes. In most the cases, the leap condition entirely depends on the nature of the process.

Emphasis was also given to formulate different algorithms to select $\tau$ efficiently. Euler tau leaping and $k_\alpha$ leaping by Gillespie [17] were explicit methods for selecting $\tau$ as these methods only depends on the status of the current stage. However, for stiff systems, they did not work well. For these type of systems, Rathinam et al. [23] proposed the implicit tau leaping method. Later, Cao et al. [9] came up with adaptive tau selection for both the implicit and explicit methods.

The tau leaping method may have accelerated the speed of simulation, but it introduced a new problem of negative population. As the number of events are approximated by Poisson random number, and Poisson random is unbounded, the approximated number of events may involve more particles than the existing number of particles in the system. To solve this, both Tian and Burrage [28] and Chatterjee et al. [10] separately proposed binomial tau leaping and Cao et al. [7] introduced the idea of critical population. However, both of these added some extra computational overhead.

The analysis of the error and stability of these approaches have been studied by Cao et al. [8], Rathinam et al. [24] and Anderson et al. [3].
All these algorithms and methods tried by different people include selection of $\tau$ before a leap is made. Thus, this basic tau leaping method by Gillespie [17] and all its modifications are known as pre leap method. However, in numerical simulations post checking provides better accuracy, though in earlier years, post leap checking was intentionally avoided because rejection of leaps may introduce biasness in the system. If the leap is too large, then it will be rejected, and shorter leaps will be chosen. However, this is a probabilistic system, and choosing only small number of events will force the system to be at only lower side of the distribution. Thus, the result will be biased on one side. Anderson [2] came up with a solution of this problem, and introduced post leap checking neglecting the biasness by storing all the information of the rejected leaps. However, this algorithm is quite complex and it is required to store a large amount of data.

As most of the time adaptive methods are only for stochastic chemical reaction problem, in our study, we will mainly try to focus on introducing effective time adaptive algorithms for stochastic particle coagulation. The most important thing to emphasize is to select a proper leap selecting criteria for pre leap method and leap acceptance criteria for the post leap one. As discussed previously, there is already a few existing algorithms to handle this problem for stochastic chemical reaction problem. However, the pairwise nature of the events in coagulation problem makes it difficult and costly to incorporate those algorithms in our problem. Thus, we will try to investigate new and simple conditions for both pre leap and post leap methods. We will also provide theoretical argument for choosing those conditions. The numerical performance of those conditions will also be studied and compared. Then, we will try to figure out the best condition and method for coagulation problem.

In chapter 2, we will introduce our problem by stating Smoluchowski’s coalescence equation [27] and then the basic algorithm proposed by Gillespie [14] will be discussed. The numerical results using Gillespie’s algorithm will be studied for simple additive kernel in comparison with the analytical solution to exhibit the performance of that algorithm. In chapter 3, the modification of Gillespie’s algorithm, both the pre leap and the post leap method will
be discussed. This chapter will also include the idea and theoretical backgrounds used to develop this methods along with different modifications of these methods over last few years. Then we will present the new theoretical approach of this current work to select simple and suitable conditions for both pre leap and post leap method. Some of this conditions will be similar to one another, we will rule out the similar ones by presenting some theoretical claims. In chapter 4 we will present numerical results for both pre and post leap methods using the conditions derived in chapter 3. We will put on a comparative study incorporating those conditions in both the methods for two different kernels and different initial conditions. Additive and Brownian kernels are used for this study. In chapter 5, we will try a make a conclusion from the numerical evidences from 4 and then will suggest some future works that can be continued to obtain more simpler and efficient simulation.
CHAPTER 2

SMOLUCHOWSKI’S COALESCEENCE EQUATION AND STOCHASTIC SIMULATION ALGORITHM

In this chapter we will discuss about the Smoluchowski’s Coalescence Equation (SCE) which governs the coalescence process. This differential equation was originally developed by Smoluchowski [27]. Gillespie [13] showed that several approximations were needed to be assumed to come up with this simpler form of SCE. The analytical solution for SCE was later available for few specific simple kernel. However, for complicated kernels, deriving an analytical solution is quite difficult.

In 1975, Gillespie [14] proposed the Stochastic Simulation Algorithm (SSA) to solve the stochastic coalescence problem by taking account all the important correlations.

2.1 Smoluchowski’s Coalescence Equation

In the stochastic coalescence model, we need to consider the existence of a function $C(x, y)$, called the coalescence kernel. Let, $N(x, t)$ is the number of particles/droplets of size $x$ present in the cloud at a specific time $t$, and consider two droplets of different sizes $x$ and $y$ moving in a given volume of a cloud. Then through the coalescence process, they may merge into a single droplet with size $x + y$ provided that they are sufficiently close. By definition

$$C(x, y) \, dt \equiv \text{Probability that a given pair of cloud droplets}$$
$$\text{with sizes } x \text{ and } y \text{ will coalesce in the next}$$
$$\text{infinitesimal time interval } dt.$$

As the nature of the kernel is stochastic, a complete analytical solution will consist of a probability function that will give a particular droplet size distribution at any time $t > 0$. Without less approximations, a complete analytical
solution can be obtained for simple kernels like $C(x, y) = \text{constant}$. However, it requires a lot of calculations, and for complex kernels it becomes really difficult.

Another common practice for this time evolution problem is to predict the quantity $\bar{N}(x, t)$ where

$$\bar{N}(x, t) \, dx \equiv \text{average number of droplets in the cloud at time } t$$

with sizes between $x$ and $x + dx$.

Smoluchowski [27] proposed the following discrete form of Smoluchowski Coalescence Equation (SCE):

$$\frac{dN(x, t)}{dt} = \frac{1}{2} \sum_{i=1}^{x-1} N(x - y, t) N(y, t) C(x - y, y) - \sum_{i=1}^{\infty} N(x, t) N(y, t) C(x, y)$$  \hspace{1cm} (2.1)

In SCE, the first term describes the average rate of production of droplets of size $x$, due to coalescence of droplets whose sizes sum to $x$. The second term describes the average rate of depletion of $x$ droplets due to the coalescence with other droplets.

However, To investigate the reasons behind writing the SCE in above form for cloud particle growth, Gillespie [13] had to make a few assumptions. First one was, the probability of having exactly $N$ particles of size $x$ at a time $t$ does not depend on the numbers of particles of other size present at that time. The second assumption was that two identical particles will not coagulate under any circumstances.

The SCE is truly deterministic and relies on the assumptions stated previously. Because of these, any solution to the SCE will therefore always be just an approximation to the real average droplet number concentration.

There are specific methods to solve SCE analytically and numerically. The analytic solution is available for a few specific kernels. Using different ODE solvers that are now available in various software packages, one can easily obtain a numerical solution. As stated earlier, both the solutions are approx-
imate solutions. However, Gillespie [14] proposed a new stochastic algorithm to solve this coalescence process. Unlike SCE, this method is directly developed from the definition of coalescence kernel considering all the important correlations. This new algorithm is known as Stochastic Simulation Algorithm (SSA).

2.2 Analytical Solution

Studies shows that, the nature of the analytical solution of SCE is entirely dependent upon the coalescence kernel. For three specific kernels: 1, $x + y$ and $xy$, the analytical solutions can be formulated without much difficulties. The solutions for these three specific cases are given in table 2.1 (Aldous [1]).

Table 2.1: Analytical Solutions for SCE

<table>
<thead>
<tr>
<th>Kernel</th>
<th>$C'(x, y) = 1$</th>
<th>$C'(x, y) = x + y$</th>
<th>$C'(x, y) = xy$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N(x, t)$</td>
<td>$(1 + \frac{\lambda}{2})^{-2}(\frac{t}{2\pi t})^{-\frac{1}{2}}$</td>
<td>$e^{-t}B(1 - e^{-t}, x)$</td>
<td>$x^{-1}B(t, x)$</td>
</tr>
<tr>
<td>$0 \leq t &lt; \infty$</td>
<td>$0 \leq t &lt; \infty$</td>
<td>$0 \leq t \leq 1$</td>
<td></td>
</tr>
</tbody>
</table>

Where $B$ is the Borel Distribution, which is one of the offspring of Poisson Distribution.

$$B(\lambda, x) \equiv P(Z_\lambda = x) = \frac{(\lambda x)^{x-1} e^{\lambda x}}{x!} \quad x = 1, 2, 3, ..., 0 \leq \lambda \leq 1$$

2.3 Solution from ODE solvers

The Smoluchowski’s Coalescence Equation (SCE) (2.1) is a simple time-dependent ordinary differential equation. This type of equations can be solved very easily for some given simple kernels using the different subroutines that are available in commercial packages.

One thing to be considered is equation (2.1) contains an infinite sum in its right hand side. None of the commercial packages can handle that. Thus,
we have to consider finite number of particles. In our case we solved the following equation instead of equation (2.1):

\[
\frac{dN(x, t)}{dt} = \frac{1}{2} \sum_{i=1}^{x-1} N(x - y, t)N(y, t)C(x - y, y) - \sum_{i=1}^{1000} N(x, t)N(y, t)C(x, y)
\] (2.2)

In this study, MATLAB is used to carry out numerical solution. Subroutine ODE45 was used for this purpose. ODE45 mainly solves non stiff ordinary differential equations with medium order accuracy. It takes the ODE’s as function inputs and solves them according to the given initial condition. We also need to provide the time span to ensure how long the process will run. There are two types of error tolerance that can be mentioned. One is 'RelTol' which is a scaler quantity of relative error tolerance. The other one is 'AbsTol' which is a vector quantity that prescribes the magnitude of absolute error of every element of solution vector. By default 'RelTol' is set to $10^{-3}$ and 'AbsTol' for each component is set to $10^{-6}$.

2.4 The Stochastic Simulation Algorithm (SSA)

The droplet growth by collision-coalescence process is governed by the collection of small cloud droplets with large cloud droplets. If a few large cloud droplets are present in the initial distribution, they will speed up the coalescence process and will cause rapid formation of rain. The situation is quite complex. To avoid this, Bayewitz et al. [5] introduced the concept of well mixed clouds. In a ‘well mixed’ cloud, all the cloud droplets are equally likely to take part in coagulation. In reality, the clouds are not well mixed.

In developing the Monte Carlo algorithm, Gillespie has considered a well mixed cloud which evolves according to a given coalescence kernel. The nature of the kernel is predetermined and known i.e. it is not required to calculate the kernel. In the well mixed cloud, there are $N$ droplets at any time $t$. Let $X_i$ denote the size of the droplet $i$. Gillespie [13] defines the coalescence probability density function $P(\tau, i, j)$ by the following statement:
\[ P(\tau, i, j) \, d\tau = \] probability at time \( t \) that the next coalescence will occur in the time interval \( (t + \tau, t + \tau + d\tau) \), and will be the coalescence of droplet size \( i \) and \( j (i < j) \).

![Diagram showing the domain of definition of the function \( P(\tau, i, j) \) for 10 droplets, consisting of points \((\tau, i, j)\) vertically above integer lattice points in the \( i,j \) plane.](image)

Figure 2.1: The domain of definition of the function \( P(\tau, i, j) \), when there are 10 droplets, consists all points \((\tau, i, j)\) lying vertically above the darkened integer lattice points in the \( i,j \) plane (Gillespie [14]).

According to the probability theory, the function \( P(\tau, i, j) \) is a joint density function on a space of continuous variable \( \tau \) and the two discrete variables \( i \) and \( j \).

Next, coalescence kernel is defined by a set of numbers

\[ C_{ij} = C(X_i, X_j) \]

where \( i = 1, 2, \ldots, N - 1 \); \( j = i + 1, i + 2, \ldots, N \)

Here, \( i < j \) uniquely labels each of the \( N(N - 1)/2 \) distinct pairs of cloud droplets. In this case, the nature of the coalescence kernel is known, and is simply given by the relation

\[ C_{ij} = C(X_i, X_j) = X_i + X_j \]

The basic steps of stochastic algorithm are shown in algorithm 1.
Algorithm 1 Stochastic Simulation Algorithm

1: Initialize time $t = 0$. Also set the initial droplet size matrix $X_1, X_2, \ldots, X_N$ of $N$ droplets.
2: Calculate the coalescence kernels $C_{ij}$ for $N(N-1)/2$ unique droplet pairs.
3: Specify the sample times where the droplet size distribution will be recorded and also the stopping time $t_{stop}$.
4: By Monte Carlo technique, generate a random triplet $(\tau, i, j)$.
5: In the droplet mass matrix, remove the droplets of size $X_i$ and $X_j$, and insert a new droplet of size $X_i + X_j$.
6: Advance time $t$ by $\tau$.
7: if $t < t_{stop}$ then
8: if $t$ has advanced through one of the sample times then
9: Record the particle size distribution at that time.
10: end if
11: Calculate the coalescence kernel $C_{ij}$ for new droplet size distribution.
12: Return to step 4.
13: else
14: Terminate the calculation.
15: end if
16: Plot necessary figures.

To perform Gillespie’s algorithm, at first an initial condition is needed which has to be provided in the form of particle size matrix. Using the given initial condition, the kernels for all the possible coagulation event have to be calculated. Then according to the probability density function $P(\tau, i, j)$, a random triplet $(\tau, i, j)$ has to be generated which will indicate the time interval when the next event will occur and participant droplet sizes. Then the particle size matrix will be modified, and this process will be repeated until a specified time is reached.

By performing several independent realizations using the same initial droplet size distribution and Monte Carlo technique, an average picture of the droplet size distribution at some specific time interval can be obtained. As the process is stochastic, the size distribution after each independent realization will be different.
2.5 Monte Carlo Step

The most important step in the algorithm is to calculate the random triplet \((\tau, i, j)\): the Monte Carlo step. Monte Carlo refers to an extensive and diverse collection of computational procedures. When Gillespie suggests his algorithm in 1975, the computational facilities were not enough, and he had to keep an eye on the data storage capability of the computers. He also referred Marsaglia and Bray. [22] in order to generate random numbers. However, in most of the modern simulation softwares, now we have built in subroutines that generate random numbers.

The main function of the Monte Carlo step is to generate a random triplet \((\tau, i, j)\) in order to calculate the probability density function \(P(\tau, i, j)\). Gillespie introduced three different methods:

1. Full conditioning method: Here, the probability density function \(P(\tau, i, j)\) can be written as

\[
P(\tau, i, j) = P_1(\tau) \, P_2(i|\tau) \, P(j|\tau, i)
\]

where \(P_1(\tau)\) is the probability of next coagulation event at time \(t + \tau\) and \(t + \tau + d\tau\) between any two particles present in the cloud, \(P_2(i|\tau)\) is the conditional probability that one of the particles that is coagulated in the time interval is the \(i\)-th particle, and \(P_3(j|\tau, i)\) is the conditional probability that the other particle that is coagulated with the \(i\)-th particle at that time interval is the \(j\)-th particle. The expressions for calculating those probabilities are as follows

\[
P_1(\tau) = C_0 \, \exp(-C_0 \tau), \quad 0 \leq \tau < \infty
\]

\[
P_2(i|\tau) = P_2(i) = C_i/C_0, \quad i = 1, 2, \ldots, N - 1
\]

\[
P_3(j|\tau, i) = P_3(j|i) = C_{ij}/C_i, \quad j = i + 1, \ldots, N
\]

where

\[
C_i \equiv \sum_{j=i+1}^{N} C_{ij} \quad i = 1, 2, \ldots, N - 1
\]
and

\[ C_0 \equiv \sum_{i=1}^{N-1} C_i \equiv \sum_{i=1}^{N-1} \sum_{j=i+1}^N C_{ij} \]

The next step is to generate the random triplet \((\tau, i, j)\) according to the above relations, where \(\tau\) is continuous and \(i, j\) are discrete random variables.

\(\tau\) is generated from an exponential distribution of parameter \(C_0\). Exponential distribution usually is a continuous distribution which describes the time between two events. At first, we need to generate a random number \(r_1\) from uniform distribution in the unit interval and then by using the formula below we can calculate \(\tau\).

\[ \tau = C_0^{-1} \ln(1/r_1) \]

To calculate \(i\) and \(j\), we need to satisfy following two relations by generating two more random numbers \(r_2\) and \(r_3\)

\[ \sum_{i'=1}^{i-1} C_{i'} < r_2 C_0 \leq \sum_{i'=1}^{i} C_{i'} \]

\[ \sum_{j'=i+1}^{j-1} C_{ij'} < r_3 C_1 \leq \sum_{j'=i+1}^{j} C_{ij'} \]

2. Partial conditioning method: According to this method, \(P(\tau, i, j)\) is in the form

\[ P(\tau, i, j) = P_1(\tau) \, P_4(i, j|\tau) \]

where, similarly like full conditioning method, \(P_1(\tau)\) is the probability of next coagulation event at time \(t + \tau\) and \(t + \tau + d\tau\) between any two particles present in the cloud, \(P_4(i, j|\tau)\) is the conditional probability that the coagulation will take place between \(i\)-th and \(j\)-th particle.

3. First coalescence method: This method generates the tentative coalescence time for all droplet pairs according to the kernel. The smallest of those calculated times is then taken as \(\tau\) and the particles involved
in that coagulation event are taken as $i$ and $j$. It is highly likely that this method can be complex and relatively less efficient than other two methods.

We have used the full conditioning method to generate the random triplet $(\tau, i, j)$.

### 2.6 Comparison between Solutions

As mentioned earlier, this particular part of study is performed with the simple predefined coalescence kernel $C(x, y) = x + y$. To avoid complexity, the well mixed assumption is also made here. It is assumed that initially the cloud has 100 particles of size 1. With evolution of time, these water droplets coagulate with each other and create larger particles.

Here, the particles can be arranged in the cells in two ways.

- Each droplet/particle can be treated separately, and they will be identified by their cell number. The values $X_i$’s assigned at each cell will indicate the size of the particle. Then the initial distribution would look like:

<table>
<thead>
<tr>
<th>Cell</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of the particle</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

Then for simple additive kernel, we will have

$$C_{ij} = X_i + X_j$$

Where, $X_i$ and $X_j$ are the size of the droplets in $i$ and $j$-th cell ($i \neq j$).

- Droplets of same size can be treated as a group. Here the cell number will indicate the particle size and the values in each cell will indicate the
number of particles available of that size. Then the initial distribution would look like:

Table 2.3: Initial distribution of particles (particles treated as a group)

| Cell | 1 | 2 | 3 | \ldots | 100 |
|------|---|---|---|\ldots|-----|
| Number of particle | 100 | 0 | 0 | \ldots | 0 |

Then for additive kernel, we will have

\[
C_{ij} = \begin{cases} 
  N(a) (N(a) - 1) a, & \text{if } a = b \\
  N(a) N(b) (a + b), & a \neq b 
\end{cases}
\]

(2.3)

Where, \(N(a)\) and \(N(b)\) are the numbers of the droplets in \(a\) and \(b\)-th cell. Here, the cell number represents the size of the droplets.

Both the arrangements provide similar results. The influence of several parameters on simulation result: particle size, initial distribution, initial number of the particles in the cloud, number of independent realizations performed to obtain an average spectrum will be described in following paragraphs.

Figure 2.2 shows the plot obtained from the analytical solution and ODE45 solver. In both the plots, the curves are in close agreement. The dome shaped curve indicates that, with time the particle density increases, and
after reaching its maximum, it begins to decrease. Both the analytical and the ODE solver plot reach their peak at almost the same time for a specific particle size. However, the peak of particle size 5 comes after the peak of the particle size 2. It is quite obvious because initially the system had only particle size of 1, and it requires some time for the particles to grow larger. Again the peak value of particle size 2 is much higher than that of particle size 5. The rationale behind this is as the particle size grows larger, they also collapse easily by coagulating with other particles as the probability of coagulation of larger particles is higher. That is why the larger particles do not remain in the system for a noticeable amount of time, and their maximum density is not as high as of smaller particles.

In figure 2.3, the results obtained from SSA is plotted with both analytical solution and solution from MATLAB using ODE45. It can be seen that the particle density distribution for all the different particle size are dome shaped, except of particle size of 1. This is due to the initial distribution of
this study; initially there were only 100 particles of size 1. Thus, particle size of 1 has its maximum density when \( t=0 \).

The simulation result is almost in phase with both analytical and ODE45 solver results. However, there are some fluctuations in the SSA plot; the distribution curve is not as smooth as the other two methods. This is highly expected because the process is stochastic, and there should be some randomness present in the plot. On the other hand, as the particle size increases, the randomness in the simulation result increases and the distribution curve seems to be more fluctuating. Though it seems to be a source of error, but in actual case, it is just an illusion caused by magnification. As the particle size increases, the numbers in the vertical scale which represents the particle density becomes smaller, and a small deviation in a small scale is magnified largely.

Figure 2.4: Effect of Number of Particles in SSA Simulation, Comparison with Analytical Solution

The number of particles in the initial distribution plays a significant role in simulation result. Though, all the particles present in the in initial distribution are of same size, but the number of particles may vary. Figure 2.4 shows the effect of particle number on simulation result. When there are only 10 particles present in the cloud initially, the SSA distribution curve
deviates much from the analytical one. As the number of particles in the initial distribution increases, the simulation result agrees more closely with the analytical result. With 50 initially present particles, the simulation result still shows some deviation in results, but with 100 particles it matches closely with analytical one. This is due to the fact that, as the number of particle increases, their density in cloud increases. It makes the mean free path of the particle collision smaller, so the probability of coagulating two particles within some specific time increases. In reality, clouds normally have millions of particles which can be treated separately, but it will require a great amount of computational work to simulate with a large number of particles. As the number of particles in initial distribution increases further to 300, it gives much better result.

![Figure 2.5: Effect of Various Numbers of Independent Iterations in Reducing Computational Error](image)

In SSA, to achieve a better result, it is necessary to perform several independent realizations to obtain an average spectrum. As can be seen from the figure 2.5, the number of independent realizations affects the accuracy of the result significantly. Here, the error is the $L^2$ norm of analytical and simulation results for each particle size after every independent realizations. To investigate the effect of number of particles, the curve is plotted for three different initial numbers of particles.
From figure 2.5, which is the plot of error vs number of iterations in log-log graph, it is clear that the error decreases as the number of iterations increases. The decrement of error is bounded by law of large numbers which tells that as the number of iterations goes larger, the simulation result will be a more accurate approximation of true average, and the error will decrease proportionally with \( 1/\sqrt{K} \), where \( K \) is the number of iterations. As the graph is plotted in log-log scale, it can be easily verified that, the error decreases linearly with a slope of \( -\frac{1}{2} \) which supports the law of large numbers.

From figure 2.5, it is also evident that the higher number of particles in initial distribution helps to reduce the error significantly which also supports the claim made from figure 2.4. The error curve for all three initial numbers of particles follow the same linear fashion, but none of them decreases dramatically. Rather they are more or less parallel. From these curve, it can be assumed that the number of particles surely helps to reduce the error, but the rate of this decrement is entirely dependent on the number of independent iterations which is governed by the law of large numbers.

One thing that can be noticed from figure 2.5 is after some number of iterations; the error curve may become a flat one. It indicates a vital result; with a specific number of particles, it is not possible to reduce the error beyond a certain limit, no matter how many independent realizations has been performed. For 10 initial particles, the error curve becomes almost flat after 200 iterations.
CHAPTER 3

ADAPTIVE TAU LEAPING METHODS

Stochastic Simulation Algorithm (SSA) is an exact algorithm which takes account every possible occurrence of the coagulation event in a coalescence process. It tracks the detailed history of every particle in the system. It is good in one way that we can have the exact picture of the system at any time during the process, and capture the randomness. At the same time, it has some disadvantages too. For a very large system, this algorithm becomes really expensive. In real physical systems, this is often the case.

To overcome the shortcomings of SSA, Gillespie [17] proposed another algorithm which is known as 'Tau Leaping Method'. The main objective of this method was to improve the computational efficiency by sacrificing some accuracy.

3.1 Basic Idea

The basic idea behind the 'Tau leaping method' is it takes a leap on the time axis based on a prescribed condition and then calculates the number of occurrence of each possible event. The condition should be such that the change in the coalescence kernel due to the coagulation event is small or negligible. Thus if we can generate such leaps in the time axis, it will consist many possible coagulation events instead of only one (which is the case in SSA), and we can expect to have a better simulation speed. However, calculating the leap in time step may contain some additional computational overhead which may play a negative role in the simulation cost or speed.

There are basically three basic procedures for selecting tau(τ) in a tau leaping
method. They are:

- **Constant Tau Leaping**: Here, the $\tau$ remains constant throughout the simulation process. This is not an adaptive $\tau$ selection procedure and is not a good practice.

- **Pre Tau Leaping**: In this method, the $\tau$ is selected prior to the leap based on the status of the kernels at the current stage. Then the number of occurrence of each possible event is calculated.

- **Post Tau Leaping**: This is a newer $\tau$ selection procedure. Here a leap in the time step is generated based on their value of the previous step and also depending on whether the $\tau$ was accepted or rejected on that step. Then all the possible events are calculated and particle matrix size is modified. If the new particle size matrix satisfies some certain condition, then the leap is accepted. Otherwise, the leap is rejected and a lower value of $\tau$ is generated.

All these procedures will be discussed in the following sections.

### 3.2 Constant Tau Leaping

As stated earlier, this method is not time adaptive. The leap in the time axis will be predefined, and will not change with the change in coalescence kernel. This method is not a feasible one, because choosing the constant value of the leap will really be a challenge, and there is no particular algorithm to fix that value. What we can do is we can simulate the system once using the SSA, plot the time steps chosen in SSA, and then choose an approximate average value of those time steps as the constant leap value.

Figure 3.1 shows the time steps chosen by SSA vs the constant $\tau$ method. At first, constant $\tau$ chooses bigger time steps than SSA. However, after some
time the constant \( \tau \) falls behind the time steps chosen by SSA. Thus, initially this method might leap across the time axis without simulating each possible event, but at a later time, the leap would be so small that no event will take place. This is not quite reasonable, because initially there are large number of particles, and the time required for a coagulation event will be small. However, as the coagulation events take place, the number of particles in the system will reduce and the time required for the next event will be large. So choosing a constant time leap throughout the process would not be a very good idea.

For developing both types of tau-leaping methods for our coagulation problem, we will first consider a rather simple but almost similar type of problem like stochastic chemical reaction. Then we will make necessary changes to use it for the coagulation problem.

### 3.3 Stochastic Chemical Kinetics

In stochastic chemical reactions, there are some initial number of particles of different chemical species. The reaction between the species take place through some specific reaction channels. For each reaction channel there is a *propensity function* which is similar to the *coalescence kernel* for our coagulation problem. Derivations are taken from Gillespie [17].
Suppose there are \( N \) chemical species \( \{S_1, S_2, \ldots, S_N\} \), the number of molecules of different species at a time \( t \) is given by \( \mathbf{X}(t) \equiv \{X_1(t), X_2(t), \ldots, X_N(t)\} \).

To begin with, we need an initial distribution of different molecules which is given by \( \mathbf{X}(t_0) = \mathbf{x}_0 \).

\[
X_i(t) = \text{The number of } S_i \text{ molecules in the system at time } t, \\
\quad (i = 1, 2, \ldots, N)
\]

Let, there are \( M \) reaction channels through which the reactions take place. They are denoted by \( R_1, R_2, \ldots, R_M \). Now we need to define the propensity function through which the chemical reactions take place. Like the coalescence kernel we define the propensity function as

\[
a_j(\mathbf{x})dt \equiv \text{the probability, given } \mathbf{X}(t) = \mathbf{x}, \text{ that one } R_j \text{ reaction will take place in the system in the next infinitesimal time interval } [t, t + dt] \quad (j = 1, 2, \ldots, M)
\]

The basic difference between chemical reaction model and the coagulation model is, in the chemical reaction problem, the reaction can occur through only few specific channels, but in the coalescence problem, if there are \( n \) particles in the system, then there are \( \frac{n(n-1)}{2} \) possible coalescence events. Another difference is in our coagulation problem, we need to generate two random numbers to figure out the two participant particle in the possible coagulation event. However, in case of chemical reactions, we have to generate only one random number which will indicate the possible reaction. Then we have to modify our particle matrix according to the state change vector \( \mathbf{\nu} \), which indicates the change in number of a different molecules due to that reaction. If \( j \)-th reaction takes place, then the state change vector associated with this reaction is \( \mathbf{\nu}_j \), whose \( i \)-th component is defined by

\[
\nu_{ji} = \text{the change in number of } S_i \text{ molecules produced by one } R_j \text{ reaction}, \quad (j = 1, 2, \ldots, M)
\]

Now calculate the quantity

\[
a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})
\]
To apply algorithm 1 in this problem, we just need to generate two random numbers. Generating the first random number $r_1$, we can calculate the time $\tau$ for next possible reaction, and using the second random number $r_2$ we can identify the next possible reaction event $j$. $\tau$ and $j$ should satisfy following two equations

$$\tau = a_0^{-1} \ln(1/r_1)$$

$$\sum_{j'=1}^{j-1} a_{j'} < r_2 a_0 \leq \sum_{j'=1}^j a_{j'}$$

### 3.4 Pre Leap Method

In the pre leap method, the leap over the time axis is calculated prior to the coagulation events take place. By conditioning on the current status of coalescence kernels, the leap time for the next events is approximated. Then, the number of occurrence of each possible event is calculated through Poisson random numbers.

#### 3.4.1 Poisson Random Numbers

Poisson random number approximates the number of events on a certain amount of time given that those events occur at a average rate of $\gamma$. The probability of having $k$ events during this time is

$$P(X = k) = \frac{e^{-\gamma} \gamma^k}{k!}$$

This is the probability distribution function for Poisson distribution. Both the mean and variance of Poisson distribution of parameter $\gamma$ is $\gamma$.

In the special case where $k = 0$, the probability distribution function becomes

$$P(X = 0) = \frac{e^{-\gamma} \gamma^0}{0!} = e^{-\gamma}$$
which is the probability distribution function of Exponential distribution with parameter $\gamma$. Thus, in this case, the Poisson random number becomes exponential random number.

### 3.4.2 Leap Condition

In $\tau$-leaping methods, the selected $\tau$ for every leap must satisfy some condition. It should be small enough such that during $[t, t + \tau]$ the change in state will be so negligible, and thus no propensity function will experience any noticeable change. This means, for each reaction channel $R_j$, the propensity function will remain almost constant during the period $[t, t + \tau]$, and $a_j(x)dt$ will provide the information of number of firings of $R_j$ reaction channels at any time interval $dt$ within $[t, t + \tau]$. This number of firings of one reaction channel will not depend on the occurring of other reaction events.

### 3.4.3 Basic Tau-Leap Method

After generating such a $\tau$, we need to approximate the number of occurrence or firings of each reaction event. If $\{k_1, k_2, \ldots, k_M\}$ are the number of firings of each of $M$ reaction channels, Gillespie [17] shows that, the number of firings of each event can be approximated by Poisson random number of parameter $a_j(x)\tau$. Since we already know the state change vector $\nu_j$ for every possible reaction channel, and each $R_j$ reaction changes the $S_i$ population by $\nu_{ji}$ molecules, the total change in state during the time $[t, t + \tau]$ is given by

$$\lambda = \sum_{j=1}^{M} k_j(x, \tau) \nu_j$$  \hspace{1cm} (3.1)

where, the number of firings are approximated by Poisson random number.

$$k_j(x, \tau) \approx \mathcal{P}_j(a_j(x, \tau))$$  \hspace{1cm} (3.2)

Now according to Gillespie [17], the basic steps for $\tau$ leaping are

- Choose a value that satisfies the leap condition. This value should be
such that the resulting $\lambda$ will be so small that for every $R_j$, $|a_j(x + \lambda) - a_j(x)|$ is almost negligible.

- For each $j = 1, 2, \ldots, M$, generate a sample value $k_j$.
- Calculate $\lambda$.
- Effect the leap by replacing $t$ by $t + \tau$ and $x$ by $x + \lambda$.

### 3.4.4 Procedure for Selecting Tau

There are several methods proposed by several people to determine $\tau$ for pre-leap method. The method should be quick and accurate enough to determine the largest possible value of $\tau$ which satisfies the leap condition.

The first method was proposed by Gillespie [17]. As the number of firing of each reaction event $k_j$ is approximated by a Poisson random number of parameter $a_j(x)\tau$, then the expected net change in state in $[t, t + \tau)$ will be

$$\bar{\lambda} \equiv \bar{\lambda}(x, \tau) = \sum_{j=1}^{M} [a_j(x)\tau] \nu_j = \tau \xi(x) \quad (3.3)$$

where we define

$$\xi(x) = \sum_{j=1}^{M} a_j(x) \nu_j \quad (3.4)$$

Now, to satisfy the leap condition, we require the change in the propensity function for each reaction channel be small i.e. this change should be bounded by a fraction $\varepsilon$ ($0 < \varepsilon < 1$) of the sum of all propensity functions:

$$\Delta a_j(x, \tau) = |a_j(x + \bar{\lambda}) - a_j(x)| \leq \varepsilon a_0(x) \quad (3.5)$$

By approximating the left hand side of equation (3.5) by first order Taylor approximation

$$a_j(x + \bar{\lambda}) - a_j(x) \approx \bar{\lambda}. \nabla a_j(x) = \sum_{i=1}^{N} \tau \xi_i(x) \frac{\partial}{\partial x_i} a_j(x) \quad (3.6)$$
Now define

\[ b_{ji}(x) \equiv \frac{\partial a_j(x)}{\partial x_i} \]

With this, equation (3.6) becomes

\[ \tau \left| \sum_{i=1}^{N} \xi_i(x)b_{ji}(x) \right| \leq \varepsilon a_0(x) \quad (j = 1, 2, \ldots, M) \quad (3.7) \]

We need to choose the largest value of \( \tau \) which is in agreement with this condition for a given value of \( \varepsilon \)

\[ \tau = \text{Min}_{j\in[1,M]} \left\{ \left. \frac{\varepsilon a_0(x)}{\sum_{i=1}^{N} \xi_i(x)b_{ji}(x)} \right| \right\} \quad (3.8) \]

In SSA, the expected time of next reaction is \( 1/a_0(x) \). If the calculated \( \tau \) in this method is only a few multiple of \( 1/a_0(x) \), then it is better to switch back to SSA, because calculating \( \tau \) in this method introduces some computational overhead. Then, this method will be more expensive.

One can also replace \( a_0(x) \) by \( a_j(x) \) in the right hand side of (3.5). However, if the number of molecular species for any \( R_j \) reaction approaches zero, \( 1/a_j(x) \) would be very small and it will be difficult to satisfy the leap condition. It is also important to make a reasonable choice of \( \varepsilon \), because a smaller \( \varepsilon \) would lead to smaller leaps.

Gillespie and Petzold [18] came up with another \( \tau \) selecting procedure for pre-leap method. This is almost similar to the method that has been discussed previously. Equation (3.6) can be written as:

\[ \Delta a_j(x, \tau) = a_j(x + \bar{\lambda}) - a_j(x) \approx \bar{\lambda}.\nabla a_j(x) = \sum_{i=1}^{N} \bar{\lambda}_i \frac{\partial a_j(x)}{\partial x_i} \quad (3.9) \]

Here \( \bar{\lambda}_i \) is the change in \( i \)-th molecular species due to all the reactions. As the firing of all the reactions are approximated by Poisson random numbers, with the equations (3.1) and (3.2) we can write
\[ \lambda_i \approx \sum_{j'=1}^{M} \mathcal{P}_{j'}(a_{j'}(x), \tau) \nu_{ij'} \]  

(3.10)

Substituting this in (3.9), interchanging the order of two summations, and by defining

\[ f_{jj'}(x) = \sum_{i=1}^{N} \frac{\partial a_j(x)}{\partial x_i} \nu_{ij'} \]  

(3.11)

we obtain

\[ \Delta a_j(x, \tau) \approx \sum_{j'=1}^{M} f_{jj'}(x) \mathcal{P}_{j'}(a_{j'}(x), \tau) \]  

(3.12)

Now, the mean and variance of \( \Delta a_j(x, \tau) \) can be computed as

\[ \langle \Delta a_j(x, \tau) \rangle \approx \sum_{j'=1}^{M} f_{jj'}(x) \langle \mathcal{P}_{j'}(a_{j'}(x), \tau) \rangle \]  

(3.13a)

\[ \text{var}\{\Delta a_j(x, \tau)\} \approx \sum_{j'=1}^{M} f_{jj'}^2(x) \text{var}\{\mathcal{P}_{j'}(a_{j'}(x), \tau)\} \]  

(3.13b)

Since \( \langle \mathcal{P}_{j'}(a_{j'}(x), \tau) \rangle = \text{var}\{\mathcal{P}_{j'}(a_{j'}(x), \tau)\} = a\tau \), we can define

\[ \langle \Delta a_j(x, \tau) \rangle \approx \sum_{j'=1}^{M} f_{jj'}(x) \langle \mathcal{P}_{j'}(a_{j'}(x), \tau) \rangle \equiv \mu_j(x)\tau \]  

(3.14a)

\[ \text{var}\{\Delta a_j(x, \tau)\} \approx \sum_{j'=1}^{M} f_{jj'}^2(x) \text{var}\{\mathcal{P}_{j'}(a_{j'}(x), \tau)\} \equiv \sigma_j^2(x)\tau \]  

(3.14b)

Both the mean and variance should be bounded by the quantity \( \varepsilon a_0(x) \) to satisfy the leap condition.

\[ |\mu_j(x)\tau| \leq \varepsilon a_0(x) \quad \text{and} \quad \sigma_j(x)\tau^{1/2} \leq \varepsilon a_0(x) \]  

(3.15)

Thus we come up with the following new \( \tau \) selecting procedure

\[ \tau = \min_{j \in [1,M]} \left\{ \frac{\varepsilon a_0(x)}{|\mu_j(x)|}, \frac{\varepsilon^2 a_0^2(x)}{\sigma_j^2(x)} \right\} \]  

(3.16)

The first term inside the parenthesis of right hand side of (3.16) is similar to the condition derived in (3.8). However, computing the 2nd term in the
parenthesis of (3.16) introduces some additional computational overhead. If there are \( M \) reaction channels, we need to calculate \( M^2 \) functions to calculate the quantity in (3.11) and then \( 2M \) functions for calculating the quantities in (3.14). If \( M \) is very large, then this overhead is really expensive.

The two alternative bounding procedure in (3.16) are not equivalent to each other for a given value of \( \varepsilon \). Also, neither procedure is more correct than the other one.

Both the methods described previously allow relatively small propensity functions to change by a relatively large amount as the change in each propensity function is bounded by a fraction of sum of all the propensity functions. This is a strong violation of the leap condition, as we require to keep this change negligible. Thus it would be helpful to use the following condition instead of equation (3.5):

\[
|a_j(x + \bar{\lambda}) - a_j(x)| \leq \varepsilon a_j(x); \quad j = 1, 2, \ldots, M
\]  

(3.17)

However, if the propensity function for a specific reaction channel \( R_j \) is small, it will force us to choose a smaller value of \( \tau \). This will be strongly against our purpose of using \( \tau \) leap method. To overcome this problem, if the minimum possible change in a propensity function is \( c_j \), then we can use the formula suggested by Cao et al. [8]

\[
|a_j(x + \bar{\lambda}) - a_j(x)| \leq \max\{\varepsilon a_j(x), c_j\}
\]  

(3.18)

Without conditioning in the propensity functions, one can also proceed in \( \tau \)-leaping method by conditioning on change in number of molecules of different species. This type of study was performed by Cao et al. [8]. The minimum possible change in number of molecular species is 1. Thus, this \( \tau \) selection procedure will select \( \tau \) such that the change in number of molecules of a species will be bounded by a fraction of total number of molecules or 1, whichever is maximum.

\[
\Delta X_i \leq \max\{\varepsilon X_i, 1\}
\]  

(3.19)
3.4.5 Problem with Negative Population

One major problem using Poisson random number to approximate the number of firings of different reaction channels is it may lead to negative population of a reactant molecule species. This case usually happens due to multiple firings of reactions that are only few molecules away to consume all the molecules of one or more of their reactants. This occurs mainly because Poisson random variable is unbounded. To avoid this, binomial tau leaping was proposed independently by Tian and Burrage [28] and Chatterjee et al. [10].

Cao et al. [8] proposed a new method for avoiding negative population of the species molecules. The decided to divide the molecules to critical and noncritical molecules according to their number present at the system at a specific time. A upper limit is set, and if the number of molecules of a specific species falls below that number, then they will be treated as critical molecules. Then we will calculate two $\tau$'s. One for non critical species using (3.16) and other for critical species using SSA. Smaller one will be considered as the new leap in the time axis.

However, in our simulation, we tried to use a simple method to deal with this problem. Suppose there are only $n$ molecules of a particular size is left to be consumed. In a specific leap time $\tau$, we found that it takes part in $m$ coagulation events. Now, if $m > n$, then the problem with negative population will arise. If such happens, we allow only $n$ coagulation events of that particle to happen, neglecting those $m - n$ extra events. Thus, if at any point, the number of particles of any size becomes $-ve$, we forcefully make it zero. It may introduce some errors in our computation, but the main thing is the number of events taking place during this leap is less than the calculated number of events during this leap. Thus, this approximation is in the conservative side and there is no chance to violate the leap condition.
3.4.6 Pre Leap Method for Coalescence Problem

In the coalescence problem, if we have \( N \) particles of \( N \) different sizes, then the number of possible coagulation events are \( N(N-1)/2 \), which is of \( \mathcal{O}(N^2) \).

If \( N \) is very large, then the number of possible coagulation events become terribly large. For computational simplicity, we start the simulation only with particles of two or three different sizes. However, as the time progresses, particles of different sizes show up in the system and the domain of definition of the system (see figure 2.1) become too large. So using equation (3.16) to calculate the leap time does not pay off. Calculating the quantity in equation (3.11) will be \( \mathcal{O}(N^4) \) then. Thus calculating the leap time for this type of system using either equation (3.8) or equation (3.16) is not a good idea. In later sections we will derive new and simple conditions for selecting \( \tau \)'s for pre leap method.

The basic steps of pre leap algorithm for stochastic coalescence problem in given in algorithm 2.

3.5 Post Leap Method

In post leap method, at first the leap is made, then number of firings of all the possible events are generated, and then a certain leap condition is checked. If the condition is satisfied, then the leap is accepted, otherwise rejected. In recent years, post leap methods for this probabilistic model was intentionally avoided due to the concern that rejected leaps might introduce some kind of bias ness. However, Anderson [2] proposed a \( \tau \) leap method that would overcome this problem.

In the post leap method proposed by Anderson [2], If a leap is rejected, still the state of the system at that time is recorded, and those data are used to calculate the state of the system at a shorter leap. Thus, the system is not forcefully made to make changes so that it can only accommodate small changes, otherwise the system will be biased.

Below we will describe the post leap method according to Anderson [2].
Algorithm 2 Pre Leap Algorithm

1: Initialize time $t = 0$. Also set the initial droplet size matrix $X_1, X_2, \ldots, X_N$ of $N$ droplets.
2: Calculate the coalescence kernels $C_{ij}$ for $N(N-1)/2$ unique droplet pairs. Then calculate $C_0$.
3: Specify the sample times where the droplet size distribution will be recorded and also the stopping time $t_{stop}$.
4: Generate the leaping time using a preferred $\tau$ selecting procedure.
5: Use Poisson random number to estimate the number of occurrence $n_{ij}$ of each possible events with $C_{ij} > 0$.
6: if $n_{ij} > 0$ then
7: Reduce the particle of size $i$ and $j$ by $n_{ij}$, and increase number of particle of size $i + j$ by $n_{ij}$
8: if The number of any of the particle becomes $-ve$ then
9: Set the number of that particle to be zero
10: end if
11: end if
12: Advance time $t$ by $\tau$.
13: if $t < t_{stop}$ then
14: if $t$ has advanced through one of the sample times then
15: Record the particle size distribution at that time.
16: end if
17: Calculate the coalescence kernel $C_{ij}$ for new droplet size distribution.
18: Return to step 4.
19: else
20: Terminate the calculation
21: end if
22: Plot necessary figures.

3.5.1 Basic Model

We will at first consider the chemical reaction system discussed in section 3.3. There are $N$ chemical species whose number of molecules is given by $X = \{X_1, X_2, \ldots, X_N\}$. At any time $t$ we have $X(t) = x$. Reactions are taking place through $M$ reaction channels. Each channel is associated with a propensity function $a_j(x)$. We also know that each reaction is associated with a state change vector $\nu_j$. If the $j$-th reaction fires $k_j(t)$ times up to time $t$, then the state of the system at time $t$ is

$$X(t) = X(0) + \sum_{j=1}^{M} k_j(t) \nu_j$$  \hspace{1cm} (3.20)
When the SSA for chemical reaction system was developed, one of the fundamental assumptions that Gillespie [15, 16] made was the probability of \( j \)-th reaction to take place in a time interval \([t, t + \tau)\) is given by \( a_j(x)\tau + \mathcal{O}(\tau^2)\), whereas the probability of more than one reaction to take place is \( \mathcal{O}(\tau^2)\).

The mathematical equivalent of the previous sentence is

\[
P(k_j(t + \tau)) - k_j(t) = 1) = a_j(x)\tau + \mathcal{O}(\tau^2) \quad (3.21a)
\]

\[
P(k_j(t + \tau)) - k_j(t) \geq 2) = P(k_j(t + \tau) - k_j(t) \geq 1, k_i(t + \tau) - k_i(t) \geq 1) = \mathcal{O}(\tau^2) \quad (3.21b)
\]

for \( i \neq j \). Now we consider unit rate Poisson process \( Y_j(.) \). A unit rate Poisson process follows a Poisson distribution of parameter 1. Suppose, \( Z \) is a Poisson process of parameter \( \gamma \), and \( Y \) is a unit rate Poisson process. These two are related by the following equation

\[
Z(\tau) = Y(\gamma \tau) = Y(T) \quad (3.22)
\]

where, \( T \) is defined as the internal time of the process. The main idea of the post leap method is it converts all the Poisson processes to unit rate process. Now the equivalent statement of (3.21) for unit rate Poisson Process is

\[
P(Y_j(T + \Delta T)) - Y_j(T) = 1) = \Delta T + \mathcal{O}(\Delta T^2) \quad (3.23a)
\]

\[
P(Y_j(T + \Delta T)) - Y_j(T) \geq 2) = P(Y_j(T + \Delta T) - Y_j(T) \geq 1, \quad Y_i(T + \Delta T) - Y_i(T) \geq 1) = \mathcal{O}(\Delta T^2); \quad i \neq j \quad (3.23b)
\]

The propensity function for each reaction \( a_j(x) = a_j(X(t)) \) which remains constant before the occurrence of next reaction. Thus for given \( X(s) \) for \( s \leq t \) we see that

\[
P \left( Y_j \left( \int_0^{t+\tau} a_j(X(s)) ds \right) - Y_j \left( \int_0^t a_j(X(s)) ds \right) = 1 \right) = a_j(X(t))\tau + \mathcal{O}(\tau^2) \quad (3.24)
\]

and the probability of more than one reaction is \( \mathcal{O}(\tau^2) \). Now, comparing equation (3.21a) and equation (3.24), we can write
Thus, equation (3.20) can be written as

\[ X(t) = X(0) + \sum_{j=1}^{M} Y_j \left( \int_0^t a_j(X(s)) ds \right) \nu_j \]  

(3.26)

Anderson [2] defined \( T_j = \int_0^t a_j(X(s)) ds \) as the internal time of the Poisson process \( Y_j \) at absolute time \( t \). Note that, there are \( M + 1 \) time frames in equation (3.26). Except the first one, which is the actual time frame \( t \), the remaining are the internal time frames presented by each of the Poisson processes.

There are two basic advantages of using post leap method. Firstly, we do not have to calculate the value of \( \tau \) beforehand, as \( \tau \) is chosen adaptively based on the acceptance or rejection of previous leap. Secondly, negative population is impossible here, as we are checking the status after the leap is made, and if there is negative population of some species, the leap is rejected. Thus, we do not have to use the expensive methods prescribed by Cao et al. [8], Chatterjee et al. [10], Tian and Burrage [28].

### 3.5.2 Concept behind the Post Leap Method

Anderson [2] developed this method based on the following two facts:

i. Each of the \( M \) internal time frames are different, and they are also different from absolute time frame.

ii. The value \( Y_j(T_2) - Y_j(T_1) \) does not depend on the state of the system \( X(t) \) for \( T(t) \leq T_1 \leq T_2 \).

The state of the system \( X(t) \), all the propensity functions \( a_j(X(t)) \), all the internal times \( T_j = T_j(t) = \int_0^t a_j(X(s)) ds \), number of firings of each channel \( k_j = Y_j(T_j(t)) \), all are usually known at an absolute time \( t \). Now, we want to proceed on time \( T, T \geq T_j \), but we do not have any information for \( Y_j(T) - Y_j(T_k) \). One leap is attempted with a predetermined \( \tau \). As the number of firings of each channel is approximated by Poisson distribution,
We generate $M$ Poisson numbers with parameter $a_j(X(t))\tau$; $j = 1, 2, \ldots, M$. Suppose the number of firings of $j$-th channel at interval $[t, t + \tau)$ is given by $N_j$. Thus we can write $Y_j(T_j + a_j\tau) = N_j + k_j$. At this point we have the new state of the system, and we can now compare with the previous state to verify that whether the leap condition is satisfied or not. If it is satisfied, the leap is accepted and we move forward again with a new $\tau$.

If the condition is not satisfied, we try a shorter leap $\tau^* < \tau$. However, as we know $Y_j(T_j + a_j\tau) = N_j + k_j$, we must use it to calculate $Y_j(T_j + a_j\tau^*)$. To condition on $Y_j(T_j + a_j\tau)$, following theorem is used.

**Theorem 3.5.1.** Let $Y(t)$ be a Poisson process with parameter $\gamma$, and let $0 \leq s < u < t$. Then conditioned on $Y(s)$ and $Y(t)$, $Y(u) - Y(s)$ has a binomial $(Y(t) - Y(s), r)$ distribution where $r = (u - s)/(t - s)$.

**Proof.** Anderson [2]: Without loss of generality, we suppose $s = 0$ and $Y(0) = 0$. Let $Y(t) = N$ and $0 < u < t$. Then

$$P(Y(u) = j|Y(t) = N) = P(Y(u) = j, Y(t) = N)/P(Y(t) = N) = P(Y(t) - Y(u) = N - j)P(Y(u) = j)/P(Y(t) = N) = \frac{e^{-\gamma(t-u)}(\gamma(t-u))^{N-j} e^{-\gamma u} (\gamma u)^j}{(N-j)! \ j! \ e^{-\gamma t} (\gamma t)^N} = \left(\frac{N}{j}\right) \left(\frac{u}{t}\right)^j \left(1 - \frac{u}{t}\right)^{N-j}$$

Theorem 3.5.1 is extensively used in developing the post leap method. Initially, at the beginning of the simulation, we do not have any stored information for the future states from rejected leaps. At that points, all the $T_j$’s are zero, and if we choose a $\tau$ to leap across the time axis, we have to generate Poisson random numbers of parameter $a_j\tau$ for all the reaction channels. As the simulation progresses, we will have some stored information from the rejected leaps. Suppose, at any particular moment of simulation, for any of the reaction channels, we have stored internal times $T_1, T_2, \ldots, T_d$ and also the unite rate Poisson values $Y_j(T_1), Y_j(T_2), \ldots, Y_j(T_d)$. The last internal time when a leap was accepted for that channel is $T_j$ and $T_1, T_2, \ldots, T_d > T_j$. Now, if we want to take a leap $\tau$, $T_j + a_j\tau$ will be either greater than $T_d$ or
will fall between any of the two stored internal times.

If $T_j + a_j \tau \geq T^d$, as we already have stored information up to internal time $T^d$, we will only generate the Poisson random number of parameter $T_j + a_j \tau - T^d$, and then will add it with $Y_j(T^d) - k_j$ to find the number of firings in the interval $[T_j, T_j + a_j \tau]$.

If $T_j + a_j \tau$ is in between two stored internal times $T^i$ and $T^{i+1}$, then we have to use theorem 3.5.1, and we have to condition on both $Y_j(T^i)$ and $Y_j(T^{i+1})$ to choose a proper binomial distribution. Then, similarly like the previous one, we have to add that binomial random number with $Y_j(T^i) - k_j$ to obtain the number of firings in the interval $[T_j, T_j + a_j \tau]$.

This is how all the information from the rejected leaps are used so that the sampling path is not affected by any kind of bias ness.

3.5.3 Updating $\tau$

Once a leap has been accepted or rejected, we can modify $\tau$ accordingly. At first we need an initial $\tau$ to begin with. Inspiring for SSA, we will use $\tau = 1/C_0$ for the very first step. Then, if the $\tau$ is accepted for a leap, we would like to take a bigger $\tau$ for the next leap. In that case, we will multiply $\tau$ by $q$, $q > 1$. On the other hand, if the leap is rejected, we need to have a smaller $\tau$ so that the leap can be accepted in next trial. In that case, we multiply $\tau$ by $p$, $0 < p < 1$.

3.5.4 Leap condition

Post leap method also requires to satisfy a leap condition, depending which the leap is accepted or rejected. Like the pre leap method, we will also determine suitable leap acceptance criteria in later sections.

The complete post leap algorithm is given in algorithm 3. Note that, each event in this algorithm should be associated with a two column matrix $S_j$. 

37
Algorithm 3 Post Leap Algorithm

1: Initialize time $t = 0$. Also set the initial droplet size matrix $X_1, X_2, \ldots, X_N$ of $N$ droplets. For each event $j$ set $T_j = R_j = 0$ and $S_j = [0, 0]$ for each $j$. Calculate the initial value of $\tau$ and fix the values of $p$ and $q$.

2: Set $B_j$ = the number of rows of $S_j$.

3: if $a_j \tau + T_j \geq S_j(B_j, 1)$ then

4: Set $N_j = \text{Poisson}(T_j + a_j \tau - S_j(B_j, 1)) + S_j(B_j, 2) - R_j$

5: Set row$_j$ = $B_j$

6: else

7: Find the index $I_j$ such that $S_j(I_j - 1, 1) \leq T_j + a_j \tau \leq S_j(I_j, 1)$

8: Set $r = (T_j + a_j \tau - S_j(I_j - 1, 1))/(S_j(I_j, 1) - S_j(I_j - 1, 1))$

9: Set $N_j = \text{Binomial}(S_j(I_j, 2) - S_j(I_j - 1, 2), r) + S_j(I_j - 1, 2) - R_j$

10: Set row$_j$ = $I_j - 1$

11: end if

12: if The leap condition is satisfied then

13: Update each $S_j$ by deleting all rows less than or equal to row$_j$ and shift all other rows down. Add a new first row of $[T_j + a_j \tau, R_j + N_j]$

14: Set $t = t + \tau$

15: For each $j$, set $T_j = T_j + a_j \tau$ and $R_j = R_j + N_j$.

16: Update $\tau$ according to the condition of accepted leap

17: Calculate the new particle size distribution matrix $X(t)$, and recalculate the propensity functions.

18: else

19: Update each $S_j$ by adding the row $[T_j + a_j \tau, R_j + N_j]$ between the rows row$_j$ and row$_j + 1$ (If row$_j + 1 > B_j$, just add a last row to $B_j$)

20: Decrease $\tau$ by $\tau = p\tau$

21: end if

22: if $t < t_{\text{stop}}$ then

23: if $t$ has advanced through one of the sample times then

24: Record the particle size distribution at that time.

25: end if

26: Return to step 2.

27: else

28: Terminate the calculation

29: end if

30: Plot necessary figures.
The first column will contain the internal times $T^i$’s, and the second one will store the number of firings $Y_j$ up to the absolute time.

3.6 New Approach of Selecting Leap Conditions

The pre leap method discussed in section 3.4 is called explicit tau leaping method. It is also called Euler tau leaping as the method is similar to the Euler method for solving ODE’s. Later Rathinam et al. [23] developed implicit tau leaping methods for stiff systems. Gillespie [17] also developed mid point tau leaping method. A convergence study of these different kind of tau leaping methods was performed by Rathinam et al. [24] and Anderson et al. [3]. In this section, we will not discuss those convergence studies. Rather we will try to provide the necessary theoretical background to develop new tau selecting procedures.

To begin the analysis, we will first consider equations (3.20) and (3.26). Here, the number of firings of each event during each leap is approximated by unit rate poisson process. Equation (3.25) will only be true if the propensity function $a_j(X(s))$ is constant throughout the leap. This is true for SSA only as SSA simulates only one reaction/event at a time, so the propensity function remains constant. Thus equation (3.26) is a true representation for the system path of SSA only, and SSA is exact in this sense.

However, If we try to model the tau leaping method by equation (3.26), then we are assuming that the propensity functions are still constant, which is not the case. In reality, propensity functions can change due to only one reaction, and here several reactions are taking place. Anderson et al. [3] formulated the following path wise representation for Euler tau leaping:

$$Z(t) = X(0) + \sum_{j=1}^{M} Y_j \left( \int_{0}^{t} a_j(Z \circ \eta(s))ds \right) \nu_j \quad (3.27)$$

where $\eta(s)ds = t_n$ if $t_n \leq s < t_{n+1}$ and $Y_j$’s are same as before. An approximate representation of equation (3.25) then will be
\[ k_j(t) \approx Y_j \left( \int_0^t a_j(Z \circ \eta(s))ds \right) \nu_j \]  

(3.28)

If we subtract equation (3.27) from equation (3.26), then we have

\[ X(t) - Z(t) = \left( \sum_{j=1}^M Y_j \left( \int_0^t a_j(X(s))ds \right) \nu_j \right) - \sum_{j=1}^M Y_j \left( \int_0^t a_j(Z \circ \eta(s))ds \right) \nu_j \]  

(3.29)

Now, to obtain an error expression, we have to calculate the difference between the expected values of \( X(t) \) and \( Z(t) \) which is of \( O(\tau) \) globally and \( O(\tau^2) \) locally according to Rathinam et al. [24] and Anderson et al. [3].

\[
\begin{align*}
|\mathbb{E}[X(t)] - \mathbb{E}[Z(t)]| &= \left| \mathbb{E}\left[ \sum_{j=1}^M Y_j \left( \int_0^t a_j(X(s))ds \right) \nu_j \right] - \mathbb{E}\left[ \sum_{j=1}^M Y_j \left( \int_0^t a_j(Z \circ \eta(s))ds \right) \nu_j \right] \right| = O(\tau)
\end{align*}
\]

(3.30)

Thus, if we reduce \( \tau \), then the error will be reduced, because as we are reducing \( \tau \) we are ignoring the possibility of multiple firings of different reaction events. In that case, the propensity functions \( a_j(X(s)) \) can be treated as almost constant and we will approach towards exact SSA. Thus, if \( \tau \to 0 \), then the leaping method converges. This is why the leap condition is such as in equation (3.5).

However, in our simulation we will try to develop a much simpler leap conditions which are easy and less time consuming to calculate. To derive the leap conditions, we will convert this problem to a constrained optimization problem.

Suppose we want to minimize the error in equation (3.29) after \( n \) time steps. These time steps are given by \( \tau_1, \tau_2, \ldots, \tau_n \). At the same time we want to reach at a specific time \( T \) using those time steps. If \( e_k(\tau_k) \) is the error in the \( k \)-th step, then our constrained optimization problem is to

\[
\text{minimize } E(\vec{\tau}) = e_1(\tau_1) + e_2(\tau_2) + \ldots + e_n\tau_n
\]
subjected to \( g(\vec{\tau}) = \tau_1 + \tau_2 + \ldots + \tau_n = T \)

We need to use Lagrange multipliers, and for that we have to figure out the error functions \( e_k(\tau_k) \)'s. Now suppose, before the leap, at any time \( t \) the sum of the kernel at any time is \( C_0(t) \), and after the leap, it is \( C_0(t + \tau) \). Using the taylor expansion, we can write

\[
C_0(t + \tau) = C_0(t) + \frac{\partial C_0(t)}{\partial \tau} \tau + \mathcal{O}(\tau^2) = C_0(t) + \alpha \tau + \mathcal{O}(\tau^2) \tag{3.31}
\]

We can write equations (3.26) and (3.27) in the following simpler deterministic form assuming \( t = 0 \)

\[
X(\tau) = X(0) + \int_0^{\tau} C_0(s)ds \tag{3.32a}
\]

\[
Z(\tau) = X(0) + C_0(0) \tau \tag{3.32b}
\]

Subtracting equation (3.32b) from equation (3.32a), we get the error expression

\[
e(\tau) = X(\tau) - Z(\tau) = \int_0^{\tau} C_0(s)ds - C_0(t) \tau
= (C_0(0) \tau + \frac{1}{2}\alpha \tau^2) - C_0(0) \tau = \frac{1}{2}\alpha \tau^2
\tag{3.33}
\]

We define a new function \( L(\vec{\tau}, \lambda) \) such that

\[
L(\vec{\tau}, \lambda) = E(\vec{\tau}) - \lambda (g(\vec{\tau}) - T) \tag{3.34}
\]

where \( \lambda \) is a scaler constant. Differentiating equation (3.34) with respect to \( \tau_k \)'s and \( \lambda \), we get

\[
\frac{\partial L}{\partial \tau_k} = 0 \Rightarrow \frac{\partial e_k}{\partial \tau_k} - \lambda = 0 \tag{3.35}
\]

and

\[
\frac{\partial L}{\partial \lambda} = 0; \tag{3.36}
\]
Putting the value of \( e_k \) from equation (3.33) in equation (3.35), we get

\[
\frac{\partial e_k}{\partial \tau_k} - \lambda = \alpha_k \tau_k - \lambda = 0
\]

\[
\Rightarrow \tau_k = \frac{\lambda}{\alpha_k}
\]

(3.37)

However, sometimes it is difficult to calculate the rate of change of \( C_0 \) directly as we do not know the value of \( C_0(t + \tau) \) beforehand. To calculate \( \alpha \) indirectly, we can go back to equation (2.3) and rewrite that in the following simpler form:

\[
C_0(t) \approx KN(t)^2
\]

(3.38)

Using Taylor expansion, we can write

\[
C_0(t + \tau) \approx KN(t)^2 + 2KN(t)\tau + O(\tau^2)
\]

(3.39)

Comparing equation (3.31) and (3.38), we get

\[
\alpha \approx 2KN(t)
\]

(3.40)

Thus, using equation (3.37) and (3.40), we get

\[
\tau_k \approx \frac{\lambda}{\alpha_k} \approx \frac{\lambda}{2KN(t)} \approx \frac{\lambda N(t)}{2C_0(t)}
\]

(3.41)

### 3.7 Fixing Different Leap Conditions

Discussion in section 3.6 helps us to choose different leap conditions and leap acceptance criteria. However, we need not to perform all of those conditions. Following discussion will lead us through different tau accepting conditions. We will also figure out which of them are required to simulate.

- **Tau as a constant multiple of \( 1/C_0 \)**
  
  Equation (3.37) tells us that we can minimize the error in computation with maximized step size by choosing a \( \tau \) which is a constant multiple of...
the inverse of the rate of change of total coalescence kernel. However, in reality, it is difficult to compute the rate of change of total coalescence kernel. It also introduces more computational overhead. However, we need to calculate the total coalescence kernel in every step. We should remember that the total coalescence kernel and its derivative indicate the same growing pattern. Thus we used \( C_0 \) instead of \( \alpha \) in our first leap condition to get rid of those extra computational overheads.

1. Pre-1/\( C_0 \): We used this condition to simulate pre leap method as we need to calculate \( 1/C_0 \) in every step and this will not incorporate any additional computational cost. Thus the leap condition is

\[
\tau = \varepsilon / C_0 \quad \varepsilon > 1
\]

However, this is not a good choice as we are not using the criteria derived from (3.37).

2. Post-1/\( C_0 \): In post leap method, we can set a leap acceptance criteria that every leap should be less than \( \varepsilon / C_0 \), where \( C_0 \) value is calculated after the leap is made. However, the following argument will prove that this will not a suitable leap condition and we can choose \( \tau \)'s that will never be rejected.

Suppose at time \( t \) a leap is accepted. In post leap algorithm, if a leap is accepted we need to choose larger \( \tau \) for the next step. As there is no specific rule how large \( \tau \) to choose, we can chose \( \tau = \varepsilon / C_0(t) \), where \( \varepsilon > 1 \). This will be a bigger leap than the previous leap because \( 1/C_0 \) is increasing. Now, after the leap is made we need to chose whether or not it satisfies the leap condition \( \tau \leq \frac{\varepsilon}{C_0(t+\tau)} \). As \( 1/C_0 \) is increasing, this condition will always be satisfied. Thus no leaps will be rejected.

- Conditioning on Total Coagulation Percentage (TCP)

In this condition, the number of coagulation events is a certain percentage of total number of particles. Suppose \( N_{\text{coag}} \) is the number of coagulations and \( N_{\text{total}} \) is the total number of particles. Then the condition will be

\[
\frac{N_{\text{coag}}}{N_{\text{total}}} \leq \varepsilon \quad 0 < \varepsilon < 1 \quad (3.42)
\]
1. Pre-TCP: To use the condition of equation (3.42), we need to transform it to a $\tau$ selecting procedure. Every event has a coalescence kernel $C_{ij}$, and if we use the basics of $\tau$ leaping method, then the number of possible firings of each event can be approximated by Poisson random number of parameter $C_{ij}\tau$. As the both mean and variance of Poisson random number is equal to its parameter, we can write the expected or mean number of occurring of an event is

$$E_{ij}[C_{ij}\tau] = C_{ij}\tau$$

Taking summations over both $i$ and $j$, we can calculate the total expected coagulation events.

$$E_{\text{total}} = C_0\tau$$

(3.43)

Now, by definition $E_{\text{total}} = N_{\text{coag}}$. Thus from equation (3.43) we get

$$\frac{E_{\text{total}}}{N} \leq \frac{\varepsilon}{\tau} \Rightarrow \frac{C_0\tau}{N} \leq \varepsilon \Rightarrow \tau \leq \frac{\varepsilon N}{C_0}$$

(3.44)

Now, we fix the $\tau$ as

$$\tau = \frac{\varepsilon N}{C_0}$$

(3.45)

This is exactly the same optimized condition that we have derived from equation (3.41) where $\tau \propto N/C_0$. Thus selecting this TCP condition we are actually selecting $\tau$ for optimized condition.

2. Post-TCP: We can directly use the equation (3.42) as a leap acceptance criteria. Once the leap is made, we can easily calculate $N_{\text{coag}}$ and $N_{\text{total}}$ and then check whether the condition in (3.42) is satisfied or not. If the condition is satisfied, the leap is accepted, otherwise rejected.

- Conditioning on Individual Coagulation Percentage (ICP)
  This condition is similar to TCP, but in this case the change in number
of each particle size in conditioned.

1. Pre-ICP: Suppose we have \( i = 1, 2, \ldots, n \) different size particles and we want to use the condition in equation (3.42) for each size of particles. Following the same procedure of TCP, in this case, we will end up with \( n \) different \( \tau \)s, one for each size.

\[
\tau_i = \frac{\varepsilon N_i}{C_i}
\]

Thus we need to select \( \tau = \text{Min}[\tau_i] \), so that the condition for every particle size is satisfied.

2. Post-ICP: To use the similar condition for post leap, we need to keep in mind that problem might arise when the number of particles of any specific size becomes 0 or 1. In those cases, change in number of particle cannot be a fraction of existing number of particles. However, we should allow these events to take place, because change in number of particle by 1 is the minimum requirement for an event to happen. If \( n \) and \( n' \) represents the number of particles before and after the leap, then the condition is

\[
|n'_i - n_i| \leq \max\{\varepsilon n_i, 1\}; \quad 0 < \varepsilon < 1
\]

This condition should be satisfied for \( i = 1, 2, \ldots, N \).

- Using a constant multiple of \( 1/\alpha \) (\( \alpha = \text{Rate of Change of } C_0 \))

This is the condition we derived form equation 3.37. If we can calculate \( \alpha \), then in that case we will have the optimum value of \( \tau \) in every step.

1. Pre-\( \alpha \): To calculate \( \alpha \) we need the value of \( C_0 \) after the leap is made. However, in pre leap method, we do not know that in advance. Thus, we can not directly use equation (3.37) in pre leap, and this can not be set a \( \tau \) selecting procedure. However, we use the approximation in equation (3.41) which is derived from equation (3.37) in Pre-TCP condition. Thus, Pre-\( \alpha \) condition is an another form of Pre-TCP.

2. Post-\( \alpha \): In post leap method, the value of \( C_0 \) after the leap can
easily be calculated, thus we can use equation (3.37) as our leap acceptance criteria. However, from equations (3.39) and (3.40), we can conclude that \( \alpha \propto C_0/N \). Figure 3.2 shows the relationship between \( \alpha \) and \( C_0/N \). It is clear that, they almost show the same pattern. As the condition \( \tau \propto N \) in Post-TCP method and we can choose the controlling parameters arbitrarily, careful choice of \( \varepsilon \) for Post-\( \alpha \) method will transform it to Post-TCP method.

![Figure 3.2: Comparison of the average values of \( \alpha \) and \( C_0/N \) for 1000 runs (Brownian Kernel, 120 initial particles (100 particles of size 1, 20 particles of size 100))](image)

Thus, we used following three different \( \tau \) selecting procedures for post leap method and two different \( \tau \) acceptance criteria for post leap method.

- Pre-\( 1/C_0 \)
- Pre-TCP
- Pre-ICP
- Post-TCP
- Post-ICP

The comparison of performance of these different methods and conditions will be discussed in next chapter.
CHAPTER 4

NUMERICAL RESULTS

In this chapter, we will present a comparative study of different tau leaping methods using the newly developed conditions. Our main objective was to stochastically simulate the equation (2.1) with different tau leaping methods to earn better simulation speed. In this chapter, we will demonstrate the variation of time steps chosen by different tau leaping methods for different kernels and initial conditions. Also the error vs cost plots of different tau selecting conditions using different kernels and initial conditions will also be shown.

4.1 Kernels and Initial Conditions

In this study we have used two different kernels: Additive kernel and Brownian kernel. Again, we simulate the additive kernel for one initial condition, and the Brownian kernel for two initial conditions.

4.1.1 Additive Kernel and Initial Conditions

Additive kernel is the most simple kernel, though its not a realistic one. The additive kernel involving particles $i$ and $j$ is given by

$$C_{ij} = \begin{cases} N(i)(N(i) - 1)i & \text{if } i = j \\ N(i)N(j)(i+j) & \text{if } i \neq j \end{cases}$$

(4.1)

There were 100 particles of size 1 as the initial condition.
4.1.2 Brownian Kernel and Initial Conditions

Brownian kernel is the most realistic physical kernel, though it is not as simple as the additive kernel. The volume rate of coagulation $B_{ij}$ involving particles $i$ and $j$ is given by

$$B_{ij} = \begin{cases} N(i)(N(i) - 1)R_{ij} & \text{if } i = j \\ N(i)N(j)R_{ij} & \text{if } i \neq j \end{cases}$$

(4.2)

where,

$$R_{ij} = \frac{2kT(D_i + D_j)^2}{3\mu D_i D_j}$$

$k =$ Boltzmann Constant,
$T =$ Temperature,
$\mu =$ Viscosity of air,
$D_i =$ Diameter of particle $i$,

As we know the value of $B_{ij}$, we can obtain the value of coalescence kernel $C_{ij}$,

$$C_{ij} = \frac{B_{ij}}{\text{Volume}}$$

In the first initial condition, there was 100 particles of size 1 and 10 particles of size 100. In the second one, we just doubled the number of particles of both size 1 and size 100.

4.2 Time Steps Chosen by Different Tau Leaping Methods

It is important to know how different adaptive tau leaping methods choose their time steps. The method that make bigger leaps in the time axis by maintaining a certain error limit will earn more simulation speed with better accuracy.

Figure 4.1(a) shows that, for additive kernel, Pre-1/$C_0$ condition chooses larger time steps than Pre-TCP condition, and Pre-ICP chooses the largest time step. Here we have small number of particles, and Pre-1/$C_0$ chooses the time steps as a constant multiple of 1/$C_0$ which is almost the same time steps
Figure 4.1: Leap Size for Different Tau Leaping Methods, Additive Kernel, 100 Initial Particles (All the particles are of same initial size) (a) Pre leap (Pre-1/C₀(ε = 2), Pre-TCP (ε = 0.02), Pre-ICP(ε = 0.02)) and (b) Post leap (Post-TCP (ε = 0.02), Post-ICP (ε = 0.4))

chosen in exact SSA. Thus, as for small number of particles, SSA works better, and so as the Pre-1/C₀ condition. In the other hand, Pre-TCP chooses almost constant time steps. However, as Pre-ICP is conditioned on particle of individual size, number of particles Nᵢ of size i is directly proportional to the sum of kernels Cᵢ involving the size i particle. Thus, Pre-ICP also chooses bigger time steps.

Figure 4.1(b) shows how post leap method chooses the time step for additive kernel. Both the post leap conditions choose the time steps almost in similar fashion. As the number of particles decrease with time, we expect the time steps to be larger (this is the case in pre leap) as the simulation progresses. However, in post leap method, the time steps increase and decrease randomly. The decrement of the time step at the beginning of the process is due to the fact that, the chosen time step fails to meet the leap acceptance criteria, thus a smaller time step is needed. However, the decrement of time step at the end of the process is mostly due to the generation of negative number of particles.

Figure 4.2 shows step size selection plots for Brownian kernel with higher number of particles (220 initial particles). Here, both Pre-TCP and Pre-ICP seem cheaper than Pre-1/C₀ for the given set of controlling parameter ε for different conditions. Pre-TCP initially chooses bigger time steps than Pre-ICP, but after sometime, the time steps chosen by Pre-ICP becomes larger. Post leap methods for Brownian kernel behave similarly like the additive one.
Figure 4.2: Leap Size for Different Tau Leaping Methods, Brownian Kernel, 220 Initial Particles (200 Particles of size 1 and 20 particles of size 100) (a) Pre leap (Pre-$1/C_0$($\varepsilon = 2$), Pre-TCP ($\varepsilon = 0.02$), Pre-ICP($\varepsilon = 0.02$)) and (b) Post leap (Post-TCP ($\varepsilon = 0.04$), Post-ICP ($\varepsilon = 0.4$))

The only difference is, in case of Brownian kernel, they choose much larger time steps than $1/C_0$.

4.3 Error Vs Cost Plots

In this study, we compared the error vs cost curve for different tau leaping algorithms with different leap accepting criteria. To calculate the error, the exact solution or the analytical solution is needed. The deviation of the simulation results from the analytical one will give the exact error. However, for most of the kernels, there is no analytical solution. Thus, to get an error estimate, we used equation (3.30) and calculated the $L^2$ norm of the simulation result and the exact SSA solution. As it is a stochastic simulation, we used 1000 independent realizations and then calculated the average distribution to come up with the expected distribution of particles. Thus the error was calculated using the equation (4.3).

\[
\|\text{Error}\|_2 = \|E_{\text{SSA}} - E_{\text{tau leap}}\|_2 = \sqrt{\sum_{i=1}^{N} \sum_{t=1}^{M} (X_{it,\text{SSA}} - X_{it,\text{tau leap}})^2} \quad (4.3)
\]

where $X_{it}$ is the number of particles per unit volume of size $i$ at a time $t$.

To get a cost estimate, only the variables that are computationally costly
were considered. In our study, we used only additive and Brownian kernel, which are not that costly. However, sometimes the kernel may be too costly to compute, that’s why we used it as a cost parameter. Besides this, generating random numbers are also costly. For pre leap, we need to generate Poisson random numbers, and for post leap, generation of both Poisson and Binomial random numbers are required. The total cost is the sum of the two costs (kernel and Poisson) for pre leap. For post leap, it the sum of those three costs (kernel, Poisson and Binomial). The average cost is the average number of evaluation of the cost after 1000 realizations. We did not include time as a cost parameter, because the codes were not written efficiently.

![Error vs Average Total Cost Plot](image)

Figure 4.3: Error vs Average Total Cost Plot for Pre-1/C_0 (\(\varepsilon = 0.6, 1.4, 1.6, 1.8\)), Pre-TCP (\(\varepsilon = 0.008, 0.01, 0.02, 0.05\)), Pre-ICP (\(\varepsilon = 0.02, 0.03, 0.04, 0.05\)), Post-TCP (\(\varepsilon = 0.02, 0.05, 0.1\)), Post-ICP (\(\varepsilon = 0.2, 0.4, 0.6\)) and Constant (\(\tau = 0.02, 0.04, 0.05\)), Additive Kernel, 100 Initial Particles (All the particles are of same initial size)

Figure 4.3 gives us an idea how costly different tau leaping methods are in terms of total cost for additive kernel. It is clear that post leaps are more costly than the pre leaps no matter what leap acceptance criteria is used. That is quite evident because in post leap methods, some leaps might be rejected, and then binomial random numbers are required to generate. Again, post leaps require a large number of data to be handled. Among the pre leap methods, Pre-1/C_0 works better than both Pre-TCP and Pre-ICP. As we have small number of particles Pre-1/C_0 almost chooses time steps like SSA which is exact. Thus in this case Pre-1/C_0 works better. Pre-TCP and Pre-ICP are almost similar in nature. Constant tau leap method behave al-
most similarly like pre leap methods. However, the problem is it is difficult to decide which constant time step should be used.

![Figure 4.4: Error vs Average Kernel Evaluation](image)

(a) Error vs Average Kernel Evaluation  (b) Error vs Average Poisson Evaluation

Figure 4.4: (a) Error vs Average Kernel Cost and (b) Error vs Average Poisson Cost for Pre-1/C_0 (ε = 0.6, 1.4, 16, 1.8), Pre-TCP (ε = 0.008, 0.01, 0.02, 0.05), Pre-ICP (ε = 0.02, 0.03, 0.04, 0.05), Post-TCP (ε = 0.02, 0.05, 0.1), Post-ICP (ε = 0.2, 0.4, 0.6) and Constant (τ = 0.02, 0.04, 0.05), Additive kernel, 100 Initial Particles (All the particles are of same initial size)

The three costs considered in the total cost are not equally expensive. Most of the physical kernels are much more expensive than generating random numbers. In figure 4.4, we separately considered the number of kernel evaluation and Poisson evaluation. Figure 4.4 provides us an insight that, if kernels are too expensive to compute, then using post leap methods will be a good idea. Post leap methods usually make bigger leaps, and even if the leap is rejected, in the next step, computing of the kernel is not required. However, pre leap method requires the evaluation of kernel at every step. Post leap method also works fine in terms of Poisson evaluation (Figure 4.4(b)) as it makes bigger leaps and when a leap is rejected, the number of events in next shorter leap is calculated by binomial random numbers which is less expensive to generate. One thing to notice about pre leap conditions is Pre-ICP requires less kernel evaluation but more poisson evaluation than Pre-TCP.

The error vs average total cost curves in figure 4.5 for Brownian kernel are almost similar to additive kernel in nature. In terms of total cost, pre leap and constant leap perform better. One thing to notice is, for Brownian kernel, all the pre leap conditions behave almost similarly, thus the curves are closer to one another in this case. The reason behind it is, for Brownian kernel, all the pre leap conditions choose almost similar time steps. There is no significant
Figure 4.5: Error vs Average Total Cost, Brownian Kernel (a) 110 Initial Particles (100 particles of size 1, 10 particles of size 100, Pre-$1/C_0$ ($\varepsilon = 1.4, 1.8, 2.2, 2.6, 3.0$), Pre-TCP ($\varepsilon = 0.008, 0.01, 0.02, 0.03$), Pre-ICP ($\varepsilon = 0.02, 0.04, 0.05, 0.07$), Post-TCP ($\varepsilon = 0.02, 0.03, 0.05$), Post-ICP ($\varepsilon = 0.2, 0.3, 0.5$) and Constant ($\tau = 2000, 3000, 4000$)) and (b) 220 Initial Particles (200 particles of size 1, 20 particles of size 100, Pre-$1/C_0$ ($\varepsilon = 1.8, 2.2, 3.0, 3.6$), Pre-TCP ($\varepsilon = 0.008, 0.01, 0.03, 0.05$), Pre-ICP ($\varepsilon = 0.02, 0.03, 0.04$), Post-TCP ($\varepsilon = 0.01, 0.02, 0.04$), Post-ICP ($\varepsilon = 0.2, 0.4, 0.6$) and Constant ($\tau = 2000, 3000, 4000$))

Figure 4.6: Error vs Average Kernel Cost Evaluation, Brownian Kernel (a) 110 Initial Particles (100 particles of size 1, 10 particles of size 100, Pre-$1/C_0$ ($\varepsilon = 1.4, 1.8, 2.2, 2.6, 3.0$), Pre-TCP ($\varepsilon = 0.008, 0.01, 0.02, 0.03$), Pre-ICP ($\varepsilon = 0.02, 0.04, 0.05, 0.07$), Post-TCP ($\varepsilon = 0.02, 0.03, 0.05$), Post-ICP ($\varepsilon = 0.2, 0.3, 0.5$) and Constant ($\tau = 2000, 3000, 4000$)) and (b) 220 Initial Particles (200 particles of size 1, 20 particles of size 100, Pre-$1/C_0$ ($\varepsilon = 1.8, 2.2, 3.0, 3.6$), Pre-TCP ($\varepsilon = 0.008, 0.01, 0.03, 0.05$), Pre-ICP ($\varepsilon = 0.02, 0.03, 0.04$), Post-TCP ($\varepsilon = 0.01, 0.02, 0.04$), Post-ICP ($\varepsilon = 0.2, 0.4, 0.6$) and Constant ($\tau = 2000, 3000, 4000$))
difference in the nature of the curves as we increase the number of initial particles in the system. The only difference is Post-TCP works better than Post-ICP if the number of particle is increased.

If we discretize the total cost in terms of kernel cost only, then from figure 4.6 we can see that both the pre leap and post leap methods are almost equally costly. There is not much significant difference between the methods and leap acceptance criteria. One thing to notice here is as the number of particle increases, Post-TCP condition starts to work better than Post-ICP in terms of kernel evaluation.

In terms of total Poisson evaluation, post leap method works better than the pre leap, which is similar to the case of additive kernel. As the number of particles in the system increases, the post leap method becomes less expensive (Figure 4.7).

From the above discussion, it is really difficult to conclude which method is better. Post leap method works better in terms of numbers of kernel evaluation for additive kernel, but for Brownian kernel shows no significant difference. In terms of Poisson evaluation, post leap method does significantly
better job than the pre leap. However, at the same time, this method requires
generation of binomial random number which introduces some additional
costs. Again, in post leap, some leaps are rejected, and huge amount of data
storage is required. This makes post leap a least obvious choice. It is also
very challenging to program.

4.4 Selecting the Controlling Parameter $\varepsilon$ in Pre Leap

From the error vs cost plots discussed in section 4.3, it can be concluded
that pre leap methods perform better than the post leap in most of the
cases. Again, Pre-1/$C_0$ condition works better than Pre-TCP and Pre-ICP
in those cases. This observation is somewhat different what we expected
from our discussion in section 3.6.

One reason why Pre-1/$C_0$ is performing better than Pre-TCP and Pre-ICP
might be we have only small number of particles in our system, and for a
small population of particle, SSA works better than any of the tau leaping
methods. SSA chooses its time steps from an exponential distribution of pa-
rameter $1/C_0$, and in Pre-1/$C_0$ we are using the similar condition to select $\tau$.
However, we are mostly interested in a system where the particle population
is too large. We know $C_0 \propto N^2$, and as $N$ increases, $1/C_0$ will decrease.
Thus we need to select larger values of the controlling parameter $\varepsilon$ to get
bigger leaps when particle population is large. However, we do not have any
idea what values of $\varepsilon$ is to choose.

Since the number of particles $N$ is incorporated to select $\tau$ in Pre-TCP and
Pre-ICP, these conditions would not suffer from this type of problems. Thus
we can expect larger step sizes as $N$ increases using the same set of con-
trolling parameters $\varepsilon$. Thus, Pre-TCP and Pre-ICP condition is expected to
work better if the system has larger number of particles though at this stage,
there is no theoretical proof which method between Pre-TCP and Pre-ICP
is better and cheaper.

Figure 4.8 supports the claim made in previous paragraph. From the figure,
it is evident that as the number of particle increases, Pre-TCP and Pre-ICP
Figure 4.8: Leap Size for Pre Tau Leaping Methods, Brownian Kernel, (a) 720 initial particles (700 particles of size 1, 20 particles of size 100, Pre-1/C_0(\varepsilon = 2), Pre-TCP (\varepsilon = 0.01), Pre-ICP(\varepsilon = 0.03)) and (b) 1520 initial particles (1500 particles of size 1, 20 particles of size 100, Pre-1/C_0(\varepsilon = 2), Pre-TCP (\varepsilon = 0.01), Pre-ICP(\varepsilon = 0.03))

select better and larger time steps than Pre-1/C_0 using the same values of \varepsilon’s.

Figure 4.9: Step Size Selection with Modified \varepsilon Value for Pre-1/C_0, Brownian Kernel, 1520 Initial Particles (1500 particles of size 1, 20 particles of size 100, Pre-1/C_0(\varepsilon = 13), Pre-TCP (\varepsilon = 0.01), Pre-ICP(\varepsilon = 0.03))

We can relate those to \varepsilon values of Pre-1/C_0 and Pre-TCP using the following discussion. Suppose, we know the initial number of particles in the system is N_i. Running the simulation once we can approximate the final number of particle N_f. Running the simulation only once will not harm because we need to run that thousand times to get an expected picture. Now we define

\[ N_{avg} = \frac{N_i + N_f}{2} \]
Now, we can transform the $\varepsilon$ value for Pre-TCP to Pre-$1/C_0$ using the following equation (4.4).

$$\tau_{\text{pre}}^{\text{TCP}} = \frac{\varepsilon_{\text{pre}}^{\text{TCP}} N_{C_0}}{C_0} \approx \frac{\varepsilon_{\text{pre}}^{\text{TCP}} N_{\text{avg}}}{C_0} \approx \frac{1/C_0}{C_0} = \tau_{\text{pre}}^{1/C_0}$$  (4.4)

Figure 4.9 shows how can we modify our $\varepsilon_{\text{pre}}^{1/C_0}$ value for larger number of initial particles using equation 4.4. Suppose we have $N_i = 1520$. After running the simulation once for brownian kernel, we obtain $N_f \approx 1060$. Thus $N_{\text{avg}} \approx 1300$. Thus form equation 4.4, the modified approximated value of $\varepsilon_{\text{pre}}^{1/C_0}$ is 13. If we use this value of $\varepsilon_{\text{pre}}^{1/C_0}$, from figure 4.9 we can see that Pre-$1/C_0$ chooses time steps which is now much closer to the time steps chosen by Pre-TCP and Pre-ICP.

Figure 4.10: Error Vs Average Total Cost for Brownian Kernel with Higher Initial Number of Particles (1520 initial particles, 1500 particles of size 100, 20 particles of size 100) for Modified $\varepsilon$ values for Pre-$1/C_0$ ($\varepsilon = 7, 14$), and regular $\varepsilon$ values for Pre-TCP ($\varepsilon = 0.005, 0.01$) and Pre-ICP ($\varepsilon = 0.02, 0.04$)

Figure 4.10 proves our claim that Pre-TCP and Pre-ICP will work better than Pre-$1/C_0$ as the particle number increases. Here, the simulation was run with 1520 initial particles, and from the figure, it is clear that Pre-TCP and Pre-ICP both are better choice in this case. Though, Pre-ICP is better than Pre-TCP, there is no theoretical evidence on behalf it. Further studies might be conducted to figure out which one of these is a better choice.
In this study, we have developed new and simple tau selecting procedures for pre leap method and tau acceptance criteria for post leap method to numerically simulate Smoluchowski's coalescence equation. These conditions help us to incorporate adaptive tau selection procedures in stochastic coalescence algorithm in more efficient way. There are already few existing time adaptive algorithms for similar type of problems like stochastic chemical reaction. However, the nature of pairwise occurrence of events in coalescence problem makes it difficult and costly to use those $\tau$ selecting algorithms in this problem. The new tau selecting conditions and criteria are selected through some mathematical derivation. With some numerical experiments, it can be easily concluded that pre leap check method works much better than the post leap one for our problem, no matter what acceptance criteria is used. Post leap method also requires a lot of data storage and handling which is also a complex procedure.

Among the selected pre leap conditions, though Pre-1/$C_0$ worked better for small number of particles, Pre-TCP and Pre-ICP started working better as particle number becomes larger. This was also predicted from our theoretical derivation. However, it is yet to be decided which one among Pre-TCP and Pre-ICP is the better one. Further studies might be performed to theoretically relate these two tau selecting procedures and decide which is more efficient.
REFERENCES


