2008-05-23

Modeling the hydrolyzing action of secretory phospholipase A2 with ordinary differential equations and Monte Carlo Methods

Zijun Lan Dozier
Brigham Young University - Provo

Modeling the hydrolyzing action of secretory phospholipase A_2 with ordinary differential equations and Monte Carlo Methods

by
Zijun Lan Dozier

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

in
Applied Mathematics

Department of Mathematics
Brigham Young University
August 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by
Zijun Lan Dozier

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	John C. Dallon, Chair
Date	William Smith
Date	Scott Grimshaw

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Zijun Lan Dozier in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	John Dallon Chair, Graduate Committee
Accepted for the Department	William Lang Graduate Coordinator
Accepted for the College	Tom Sederberg, Associate Dean College of Physical and Mathematical Sciences

ABSTRACT

Modeling the hydrolyzing action of secretory phospholipase A_2
with ordinary differential equations and Monte Carlo Methods

Zijun Lan Dozier

Department of Mathematics

Master of Science

Although cell membranes normally resist the hydrolysis of secretory phospholipase A_2, a series of current investigations demonstrated that the changes in lipid order caused by increased calcium has a relationship with the susceptibility to phospholipase A_2. To further explore this relationship, we setup ordinary differential equations models, statistic models and stochastic models to compare the response of human erythrocytes to the hydrolyzing action of secretory phospholipase A_2 and the relationship between the susceptibility of hydrolysis and the physical properties of secretory phospholipase A_2. Furthermore, we use models to determine the ability of calcium ionophore to increased membrane susceptibility.

ACKNOWLEDGMENTS

First and foremost I offer my sincerest gratitude to my supervisor, Dr. John Dallon, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Masters degree to his encouragement and expertise. One simply could not wish for a better or friendlier supervisor.

I would like to thank my wonderful parents and husband for helping me get through difficult times and for all their support, entertainment, and love.

I am grateful to the Department of Mathematics for assisting me in many different ways. I am especially indebted to Lonette Stoddard for all of her help.

Lastly, I wish to thank Dr. Shane Reese and Mike Ulrich for their assistance on programming.

Contents

1 Introduction of Biomathematics and Biology Background

1.1 Biomathematics

Biomathematics, or Mathematical Biology is an interdisciplinary field of academic study which aims at modeling biological process using mathematical techniques and tools [12]. Sophisticated mathematical results have been used in and have emerged from the life sciences. Examples are given by the development of stochastic processes and statistical methods to solve a variety of population problems in demography, ecology, genetics, and epidemiology, and most joint work between biologists, physicists, chemists and engineers involves synthesis and analysis of mathematical structures [13]. Pythagoras, Aristotle, Fibonacci, Bernoulli, Euler, Fourier, Laplace, Gauss, Riemann, Von Newmann, Einstein, Thompson, and Wiener are names associated with both significant applications of mathematics to life science problems and significant developments in mathematics motivated by the life sciences.

Although mathematics has been applied to the field of biology since its conception, the majority of major breakthroughs in Mathematical Biology have been achieved within the last fifty years. Main reasons include: the development of computer science makes it possible to analyze large data sets and perform more accurate simulations; advances in mathematical tools have been beneficial for obtaining a further understanding of biology fields [15]. Due to developments in the aforementioned areas in the past thirty years, dramatic results have been obtained by applying mathematics to the following areas of research: population dynamics, modeling cell and molecular biology, modeling physiological systems [15].

Early mathematical models in the life sciences described phenomena over broad ranges of parameter values and widely disparate time and space scales [13]. However, the accuracy of a mathematical model is limited by its level of realism, the information that is being sought, and the methods of analysis that might be available to study it. Euler's renewal theory in demography and Fisher and Kolmogorov's model of the formation of genetic clines are notable examples [12].

Over the past 30 years there has been a shift from mathematical analysis to computer simulation due mostly to improvements in computer power and accessibility [15]. This shift has made it possible to include more information in models and still derive useful insights from them. Moreover, bootstrapping and data mining procedures have introduced new computer-based methodologies that work directly with observed databases. Computers have freed investigators to explore more detailed mathematical descriptions of life. In this thesis, computer simulations using MATLAB and C++ code are applied to mathematical models to help understand systematic behaviors of an enzyme's hydrolysation process, which will provide a basic example of why mathematics has been useful to understand life science.

In Mathematical Biology, a number of mathematical techniques have been applied to model experimental processes. For example, ordinary differential equations (ODE) and partial differential equations (PDE) are the two most important tools for analyzing biological dynamic systems, such as molecule replication and cell reproduction; probabilistic evolutionary models propose a probabilistic interaction between the environmental variations and the biological evolution; while Markov chains are widely used to simulate the behavior of cellular [13]. In this paper, ODE models and Markov models will be applied to study the behavior of enzyme $sPLA_2$ on cell membranes.

1.2 Biology Background

Secretory phospholipase A_2 ($sPLA_2$), an enzyme widely used in physiological and biological experimental testing, hydrolyzes susceptible phospholipid of cell membranes, releasing fatty acids and lysophospholipids. By acting extracellularly, this enzyme can be adsorbed by the membranes of blood cells and effectively destroy the apoptotic cells' membrane [1]. In physiology, it may participate in various functions including digestion, clearing of dead or damaged cells, and membrane homeostasis, which is the property of a living organism that regulates its internal environment so as to maintain a stable, constant condition, such as defence against bacteria [8].

However, not all cell membranes are impressible to be hydrolyzed by sPLA2. Since cell membranes ordinarily resist the hydrolysis of this enzyme, the action of hydrolyzing can be fulfilled only when the resistance is relatively weak [5]. In other words, if the cell membrane is under healthy physical condition, the enzyme will not be active in the hydrolysation process. Properties that promote susceptibility include negative charge in the membrane surface, high curvature of the lipid bilayer, and heterogeneity of lipid components. Furthermore, the activity of $sPLA_2$ is extremely sensitive to the level of ordering of membrane lipids.

The differences between the physical and chemical properties of artificial bilayer and human erythrocyte membranes need to be considered in applying the principles above. To further explore this relationship, laboratory experiments using temperature as an experimental means of manipulating membrane physical properties were done to check the temperature dependence of membrane properties in the range of 20 to 50 degrees of centigrate. In the experiments, erythrocyte membranes were treated with and without ionomycin (Ca++) at a variety of temperatures, and the level of lipid

order was assessed by fluorescence spectroscopy using laurdan, which is a fluorescence dye used to observe physical properties of membranes [17]. The experiments indicate that erythrocyte membranes display significant temperature dependence of membrane properties in the range of 28 to 45 degrees of centigrate. Therefore, for biological systems, the variation of temperature changes the membrane order which will greatly influence the susceptibility of erythrocyte membranes.

Furthermore, laboratory experiments also suggest that the susceptibility to the enzyme can be induced by loading cells with calcium. In previous studies, it was observed that erythrocytes are amenable to the property of resisting hydrolyzing action of $sPLA_2$, but that upon the addition of a calcium ionophore such as ionomycin, they became susceptible [17] and [21]. In human erythrocytes, the ability of calcium ionophore to cause susceptibility depends on temperature, occurring best at about 35 degrees. From 20 to 60 degrees, the lipid packing decreased gradually, but calcium loading enhanced packing at temperatures around 20 degrees and greatly reduced packing at higher temperature.

What's more, a large number of studies conducted with $sPLA_2$ suggest that the hydrolysis of membrane lipids requires two steps. First, the enzyme adsorbs to the membrane surface. Adsorption, different from absorption, is a process that occurs when a liquid or solute (called adsorbate) accumulates on the surface of a solid or more rarely a liquid (adsorbent), forming a molecular or atomic film (adsorbate); however, the absorption process is when a substance diffuses into a liquid or solid to form a "solution". Secondly, a phospholipid of the membrane diffuses from the membrane into the active site of the adsorbed enzyme. The enzyme then takes hold of the polar hydrophilic head of the molecule, removes the head from the hydrophobic portion of the molecule, and carries the head into the solution, thus finishing the hydrolyzing action on the membrane [21]. However, the adsorption of $sPLA_2$ is sensitive

to neither temperature nor calcium ionophore treatment. Thus, when the absorption process is considered, the affection of temperature and calcium can be ignored.

However, there are still a lot of questions in the hydrolyzing process that are unresolved; for example, how the rolling action of enzyme on the surface of the membrane influences its absorption. Two independent mathematical models are ordinary differential equation model and a stochastic model will be applied to the analysis of this system.

2 Ordinary Differential Equation Model

2.1 The Michaelis-Menten Mechanism

The most fundamental form governing enzymic reactions is the Michaelis Menten (MM) mechanism [11] whereby a substrate molecule is converted to a product molecule by first reacting with an enzyme to produce an enzyme-substrate complex. This complex is unstable and so decays either back to the original enzyme-substrate pair or to an enzyme-product pair. In either case, the enzyme is released and available once again to bind with a new substrate molecule. Schematically, the reaction can be expressed as

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \overset{k_2}{\rightarrow} E + P \tag{1}$$

where *S, E, C* and *P* respectively represent the substrate, enzyme, complex and product molecules and the k_i are the *rate constants* controlling the speed of the reaction. The enzyme is said to *catalyse* the reaction as it facilitates the overall transition

$$S + E \xrightarrow{k} P$$

where k represents the rate of production of P and is in general dependent on a combination of the rate constants in (1). There are very many examples in nature of the occurrence of an enzymic reaction of the Michaelis Menten form. These include, for example, the hydrolyzing process of enzyme $sPLA_2$ on blood cell membrane, which was introduced in Section 1.2. In the next chapter, two Ordinary Differential equation models of the hydrolysation of $sPLA_2$ will be based on the Michaelis-Menten mechanism, and it will be applied to establish the stochastic model in Chapter 3. Therefore, an understanding of the kinetics of the Michaelis Menten mechanism is crucial to understanding our models.

2.2 Simple Ordinary Differential Equation Model

The simple ordinary differential equation model is derived by modifying a previous model [17]. The assumptions for the model are:

• The concentration of cells and therefore the total number of adsorption sites for the enzyme $sPLA_2$ is constant; however, in reality, the number of adsorption sites are decreasing because of the hydrolysation process. Due to difficulty of modeling, we will assume that the number is constant.

• The rolling process of $sPLA_2$ on the cell membranes is ignored.

• Only a small fraction of added enzyme absorbs to the cell surface; thus the concentration of $sPLA_2$ in solution can be approximated by the total enzyme concentration.

According to these assumptions, the model describing the action of $sPLA_2$ on the

surface of the membrane can be explained as follows.

Let E be the total concentration of $sPLA_2$ which is added to the erythrocyte solution in the beginning of the experiment. The concentration of adsorbed enzyme is denoted by E_B. At a rate k_1, the enzyme is adsorbed to the membrane surface. This process can be expressed as

$$E \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} E_B.$$

However, not all of the adsorbed enzyme will successfully hydrolyze erythrocytes: only the ones which are absorbed to the surface and have the phospholipid bound to the active site will be able to be hydrolyzed. These kinds of enzymes are denoted by E_B^S and the rate from E_B to E_B^S is k_2,

$$E_B \underset{k_{-2}}{\overset{k_2}{\rightleftarrows}} E_B^S.$$

At the same time, there exist a small portion of enzyme which, at rate k_0, are absorbed directly onto the active site (E_B^S).

$$E \underset{k_{-0}}{\overset{k_0}{\rightleftarrows}} E_B^S.$$

A large number of experiments of physiology [21] show that these three processes are actually physical hence they won't influence the chemical properties of both $sPLA_2$ and membranes.

Also, we assume that geometrically enzyme $sPLA_2$ is cubic, with one active side A, one top side T which is opposite to the active side, and four other sides S. Suppose

the probabilities that $sPLA_2$ lands on each of these 6 sides are equal, and let E_A, E_T, E_S, and E_A^S denote the concentration of enzyme on side A, T, S, and side A with bounded lipids, respectively.

From the reactions above, we can conclude a system of differential equations for the concentration of each substance in terms of the concentrations of all others:

$$\frac{dE}{dt} = 0$$

$$E_{sol} = E - E_A - E_S - E_T - E_A^S$$

$$\frac{dE_A}{dt} = -k_2 E_A + k_{-2} E_A^S + k_1 E_{sol}$$

$$\frac{dE_T}{dt} = -k_{-3} E_T + k_3 E_{sol}$$

$$\frac{dE_S}{dt} = -k_{-4} E_S + k_4 E_{sol}$$

$$\frac{dE_A^S}{dt} = -k_{-2} E_A^S + k_2 E_A + k_0 E_{sol} - k_{-0} E_A^S$$

$$\frac{dP}{dt} = k_{cat} E_A^S (1 - P).$$

All of the reactions above are actually preparations of the hydrolyzing action; thus, after the enzyme is adsorbed by the surface with ordered lipid, the hydrolyzing process can be continued. Let P be the product of hydrolyzing process, and k_{cat} be the rate constant of hydrolysis, thus the rate of hydrolysis ($\frac{dP}{dt}$) is given by the following:

$$\frac{dP}{dt} = k_{cat} E_B^S (1 - P).$$

According to the ODE model presented above, Figure 1 is obtained. This figure illustrates the change in the concentration of product P as the length of time in-

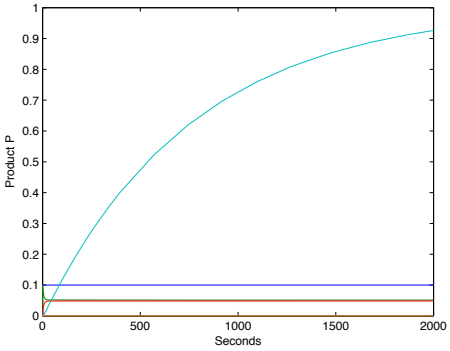


Figure 1: Simple ODE Model: In the graph, green curve represents the enzyme which are absorbed to the membrane; yellow line shows the total amount of enzyme on the membrane; while blue curve is the final product P of the hydrolyzing process. All of the models in this chapter are processed by a MATLAB function *ode15s* (see Appendix 5.1), and they all share the same legend.

creases. In the first 200 seconds of the simulation, a sharp increase in the value of P indicates that a large amount of hydrolyzing processes occurred; while after that, the gradual leveling off of the curve is approaching a steady state. From the non-rolling assumption of this model, the previously stated equilibrium generally comes from the fact that when all of the actively absorbed enzyme finished hydrolyzing the available lipid, no more reactions can occur. The results of this simulation correlate with the experimental results [17]. It is important to note that the enzyme $sPLA_2$ remains constant throughout the simulation.

From Figure 1 to Figure 3, it is easy to conclude that the amount of enzyme that is added into the solution is related with the reaction speed and product amount. More enzyme will result a higher reaction speed, but not an obvious increase in the product amount. This is due to the limited amount of reactant.

According to the first assumption of this ordinary differential equation system, the total number of absorption sites for the enzyme is constant. The equilibrium

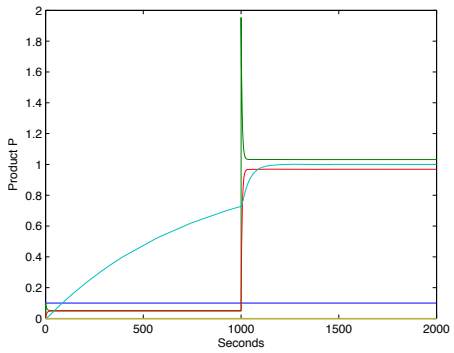


Figure 2: Simple ODE Model: The same initial amount of enzyme is added, and an extra amount (same as the initial amount) of enzyme is added at 1000 seconds.

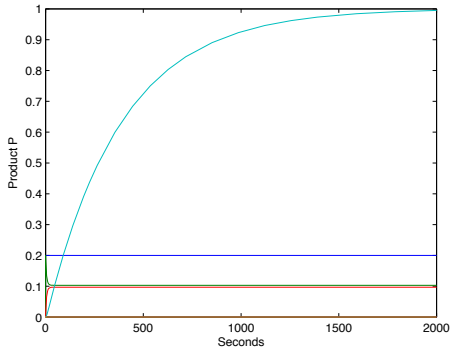


Figure 3: Simple ODE Model: Twice the initial amount of enzyme as in the simulation in Figure 1 is added in the beginning of this simulation.

constants (K_1 and K_2) are defined as follows:

$$K_1 = \frac{E_B}{E} = \frac{k_1}{k_{m1}}$$

$$K_2 = \frac{E_B^S}{E_B} = \frac{k_2}{k_{m2}}$$

The value of all reaction coefficients can be found in Appendix 5.1 [17].

From the second assumption, the total enzyme $sPLA_2$ can be approximated by the enzyme concentration E_T. Combining equations above, an explicit description of the initial hydrolysis rate can be obtained:

$$\frac{dP}{dt} = \frac{\alpha E_T K_1 K_2}{1 + E_T K_1 + E_T K_1 K_2}$$

where α is proportional to k_{cat} [17], [4] and [21].

2.3 ODE Model with Rolling Process

In order to model the rolling process of $sPLA_2$, we assume that geometrically enzyme $sPLA_2$ is cubic, with one active side A, one top side T which is opposite to the active side, and four other sides S. Also, suppose the probabilities that $sPLA_2$ lands on each of these 6 sides are equal, and let E_A, E_T, and E_S denote the event of enzyme's landing on side A, T, and S, respectively. We assume the enzyme always rolls and do not stay on one side. The rolling rate of enzyme $sPLA_2$ on the cell membrane is r. The relationship between these three type of sides are as below:

$$E_A \underset{0.25r}{\overset{r}{\rightleftarrows}} E_S,$$

and

$$E_T \underset{0.25r}{\overset{r}{\rightleftarrows}} E_S.$$

Remark: If the enzyme lands on A, it can only roll to side S by 1 rolling step, so the rate from A to S is r; however, if it lands on side S it may roll to side A, side T, or other two sides of S; thus, the rates from S to both T and A are $0.25r$.

From the relationship above, we can derive the ODE system with rolling process:

$$\frac{dE}{dt} = 0$$

$$E_{sol} = E - E_A - E_S - E_T - E_A^S$$

$$\frac{dE_A}{dt} = -k_2 E_A + k_{-2} E_A^S + k_1 E_{sol} + 0.25 r E_S - r E_A$$

$$\frac{dE_T}{dt} = -k_{-3} E_T + k_3 E_{sol} - r E_T + 0.25 r E_S$$

$$\frac{dE_A^S}{dt} = -k_{-2} E_A^S + k_2 E_A + k_0 E_{sol} - k_{-0} E_A^S$$

$$\frac{dP}{dt} = k_{cat} E_A^S (1 - P)$$

in which E is the total number of enzyme that we put in the culture, E_{sol} is the enzyme which dissolves in the solution, and E_A^S represents the enzyme actively absorbed by the cell membrane. From this system, we obtain Figure 4, which should be compared with Figure 1.

In Bell's experiment [4], there are a few very interesting results coming from the following experiments: under the same condition,

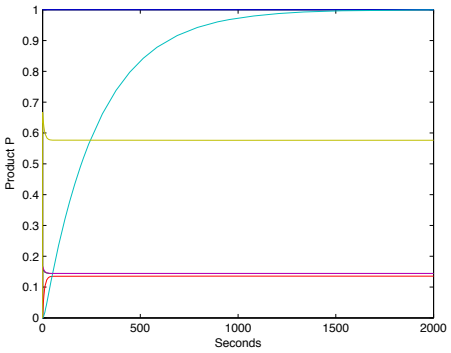


Figure 4: Rolling ODE

1. Twice the amount of enzyme solution is injected in the beginning of the experiment;

2. One amount of $sPLA_2$ is added in the beginning and an extra amount is added in the middle of the reaction.

Comparing with the experiment which starts with only one amount of initial enzyme solution, the reaction of the experiment with initially double the amount of enzyme went to the steady state faster. However, the concentration of product after the reaction stopped is about the same [17]. The reaction speed of the second experiment is not as fast as the first one, but still no optional artifact occurred upon the addition of the enzyme [4], [17]. One possible explanation of these results are: due to the crowding of enzyme on the membrane, the rolling process will be hard to occur; thus, when more enzyme is added into the solution, there will not be a large amount of enzyme in the solution that can land on the blood cell membrane. This explanation will be addressed in the Stochastic Model.

In order to keep the same experimental condition, the relationship between the variables were kept the same; however, the initial value of E was set up as 2 units for the simulation shown in figure 5 and 1 unit for the simulation shown in figure 6. In the simulation shown in figure 6, after 500 seconds (the middle of the simulation), the value of E was increased by 1 unit, which was the same as its initial value. After this alteration, the simulation was continued.

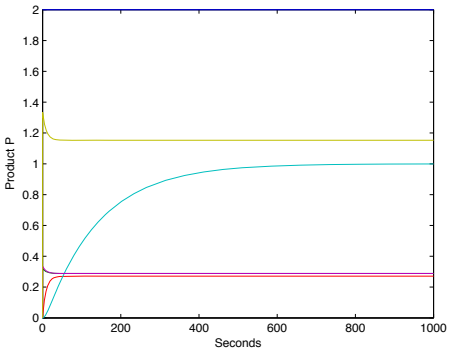


Figure 5: Double initial amount of E: In this graph, the changes of product P is shown when twice the amount of enzyme is added initially.

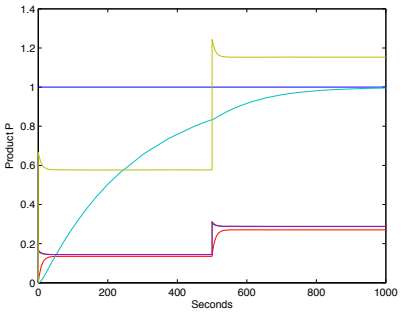


Figure 6: Additional E added in the middle of the reaction: In the middle of the reaction, the same amount of enzyme as the initially value is added.

Figure 5 and Figure 6 show the changes in the concentration of product P as the increase of the length of time. In figure 6, the curve had a sharp increase at 1000, when an additional 1 unit of enzyme was added; however, the increasing rate was not as large as the beginning of the simulation. In the end, the Product P curve ended to a steady state only about 1 unit. Thus, even though the amount of enzyme in the solution is doubled in the middle of the experiment, the amount of product P does not increase. This results comes from the property of the binding process of enzyme to the blood cell membrane. Although there was twice as much enzyme in the solution that can bind to the membrane to hydrolyze the lipids, due to the prerequisite of the reaction, not all of the hydrolysation process will be able to complete. The

prerequisite of the reaction can be interpreted as the self-protection mechanism of cell membrane. However, the exposure of erythrocytes to calcium at low levels, and to nickel or manganese with ionomycin, amplified the maximum rate of hydrolysis of the erythrocyte membrane by $sPLA_2$ [21].

Rolling is an important property of enzyme $sPLA_2$ [4]. This process allows more lipids to be hydrolyzed and more product P be made. Also, because of this rolling process, the time that the simulation needs to obtain equilibrium is longer than that is needed in a simple ODE simulation. The time that is concluded in a rolling process correlates with the experimental results. What's more, the discrepancies that occurred need to be illuminated to achieve a more precise understanding of the hydrolysation of enzyme $sPLA_2$. Therefore, in the next chapter, Markov Chain and Monte Carlo methods will be applied to perform further analysis in the hydrolysation process of $sPLA_2$.

3 Stochastic Model

3.1 Probability Theory and Stochastic Process

As a systematic mathematical study, probability emerged in the 17th century. The traditional beginning of modern probability theory is the exchange of letters in July and October 1654 between Blaise Pascal (1623-1662)and Pierre Fermat (1601-1665), two French mathematicians. The letters were written in response to the following problem: Two players, A and B, each stake 32 pistoles on a three-point game. When A has 2 points and B has 1 point, the game is interrupted and cannot continue. How should the stakes of 64 pistoles be fairly distributed? Pascal divided

the solution into two parts. Whatever the outcome of the game, A should have at least one-half of the total 32 pistoles. Therefore, the uncertain expectation concerned only the other half and A had a 50 percent chance of winning that. Therefore, the fair distribution would be that A received 48 pistoles (the 32 and one half the uncertain 32), and B received 16 pistoles. [18]

Huygens tract remained the only text on probability for 50 years. The early years of the 18th century witnessed a series of publications on probability by Montmort, Nicolaus Bernoulli, DeMoivre and posthumously Jacob Bernoulli. This might have been stimulated by 'whispers' and writings about that elusive piece Ars Conjectandi, on which Jacob Bernoulli had been brooding for 20 years, and which was still not finished when he died. After Montmort died, it was DeMoivre who reigned suppreme with his Doctrine of Chance. From the middle of the 18th century the combination of observations became an important topic that was studied by Boscovich, Laplace and others. [20]

The basic notion of probability theory is that of the *random experiment*: outcomes of an experiment (real or conceptual, but capable of being repeated,) are called *events*. The collection of all events of an experiment is called the *sample space* Ω, the points of which are the simple events. The relations between the events make new events out of those given. An event A is said to occur if and only if the observed outcome ω of the experiment is an element of the set A. Take the rolling process of enzyme $sPLA_2$ as an example; it is assumed that the shape of $sPLA_2$ is a cube with one top side (side T), one active side (side A) and four other sides (side S). When the rolling procedure of $sPLA_2$ on the blood cell membrane is considered as a random experiment, the side that the enzyme rolls onto will determine the event; the sample

space Ω is A, T, S, or the collection of all events.

Corresponding to our intuitive notion of the chances of an event occurring, we first of all review a few definitions regarding basic stochastic simulations and Markov chains [5].

Definition. Let Ω be a sample space and p a function which associates a number with each event. Then p is called a probability measure provided that
(a) for any event A, $0 < p(A) < 1$; (b) p(Ω)= 1; (c) for any sequence A1, A2,... of disjoint events,

$$p(\bigcup_i A_i) = \sum_i p(A_i)$$

By axiom (b), the probability assigned to Ω is 1. If a statement holds for all ω in a set A with p(A)=1, then it is customary to say that the statement is true *almost surely* or that the statement holds for *almost all* $\omega \in \Omega$.

Suppose we are given a sample space Ω and a probability measure p. Most often, especially in applied problems, we are interested in functions of the outcomes rather than the outcomes themselves, i.e. in our example, we are interested in the hydrolysation of the membrane, not the rolling process; however, by considering the rolling of the enzyme, we can obtain a better understanding of the membrane's hydrolysation.
Definition. A probability space $(\Omega, \mathcal{F}, P)$ is a measure space with a probability measure P.

Definition. A *random variable* X with values in the set E is a function which assigns a value $X(\omega)$ in E to each outcome ω in Ω.

A *stochastic process* with *state space* $\mathcal{E}$ is a collection $X_t; t \in T$ of random variables X_t defined on the same probability space and taking values in E. The set T is called its *parameter set*. If T is countable, especially if T = $\mathbb{N}$ = 0,1,..., the process is said to be a *discrete parameter* process. Otherwise, if T is not countable, the process is said to have a *continuous parameter*. It is customary to think of the index t as representing time, and then one thinks of X_t as the *state* or the *position* of the process at time t.

3.2 Markov Chain

In 1907, A. A. Markov started working on a chance process, in which the outcome of next experiment is only influence by the outcome of the current experiment. This process is called a Markov chain [2]. In this section, the rolling states of $sPLA_2$ will be considered as a sequence of experiments and Markov chain will be applied to imitate the hydrolyzing process.

Let $X = X_1, X_2, ...$ be a random process in the discrete state space S. It is called a Markov chain if the conditional probabilities between the outcomes at different times satisfy the *Markov property*, which we now explain.

Definition. The sequence $X_1, X_2, ...$ of S-valued random variables is said to have the Markov property if

$$\mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t, ..., X_1 = x_1) = \mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t)$$

for every sequence $x_1, ..., x_t, x_{t+1}$ of elements of S and for each time $t \geq 0$

A sequence of random variables with the Markov property is called a *Markov chain.*

Definition If $X_1, X_2, ...$ is a Markov chain, and i and j are states in S, the conditional probability

$$p_{ij}(t) \equiv \mathbb{P}(X_{t+1} = j \mid X_t = i)$$

is called the transition probability from i to j at time t. If the transition probabilities do not depend on time, we write them simply as p_{ij}, $i, j \in$ S and we say that Markov chain is *time-homogeneous.*

3.2.1 Transition Matrix

In all of the tools related to Markov chain, transition matrix is one of the most important in analyzing discrete situation. Suppose that the state space S is finite and let us write it as $S \equiv 0, 1, ..., s$. Given a set of transition probabilities, it is often useful to collect them in a matrix,

$$\mathbb{P} = \begin{pmatrix} p_{00} & p_{01} & p_{02} & \cdots & p_{0s} \\ p_{10} & p_{11} & p_{12} & \cdots & p_{1s} \\ & & \cdots & & \\ p_{s0} & p_{s1} & p_{s2} & \cdots & p_{ss} \end{pmatrix} \tag{2}$$

Notice that the row [p_{i0},p_{i1},...,p_{is}] represents all the transition probabilities out of state i. Therefore, the probabilities in the row must sum to 1. Such a square array is called the *matrix of transition probabilities*, or the *transition matrix.*

It is easy to see that the rolling process of $sPLA_2$ from one side to another de-

pend only on the present state, not the previous one. Therefore, it is reasonable to consider the rolling procedure as a Markov chain with 3 states. However, in order to apply transition matrix on the the rolling process of $sPLA_2$, several assumptions should be made.

• In this paper, we assume that the shape of the enzyme is cubic with size $45\mathring{A} \times 45\mathring{A} \times 45\mathring{A}$, with one active side A, four other sides S and one top side T which is opposite to the active side. The three dimensional structure of enzyme $sPLA_2$ resemble a flattened ellipsoid with approximate dimensions of $45\mathring{A} \times 30\mathring{A} \times 20\mathring{A}$ $(1\mathring{A} = 10^{-10}m)$[6]; however, due to the difficulties of modeling, a cube shape will be applied.

• When $sPLA_2$ lands on the cell membrane, the probabilities that it comes down on each of these 6 sides are equal. Let $\mathcal{S}$ be the sample space of enzyme's one-step rolling process, containing S, A, T as outcomes; also, S, A, T can be used as the notation of different states in the transition matrix. Therefore, p_{ij}, i, j, $\in \mathcal{S}$ shows the transition probability from state i to j.

• The enzyme is restricted to an upward, downward, leftward, or rightward rolling direction, or stays on its position.

• When all of the nearest spots are occupied, the rolling process will not be able to continue; thus the enzyme will stay in its current state, which means

$$\mathcal{P}_{\mathcal{AA}} = \mathcal{P}_{\mathcal{SS}} = \mathcal{P}_{\mathcal{TT}} = 1$$

$$\mathcal{P}_{\mathcal{AS}} = \mathcal{P}_{\mathcal{AT}} = 0$$

$$\mathcal{P}_{\mathcal{SA}} = \mathcal{P}_{\mathcal{ST}} = 0$$

$$\mathcal{P}_{\mathcal{TA}} = \mathcal{P}_{\mathcal{TS}} = 0$$

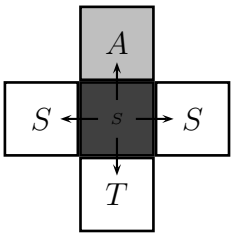
Let θ be

$$\theta = \frac{\text{Number of occupied spots}}{4}.$$

Let $\theta = 0$ if all spots are available, while $\theta = 1$ represents all surrounding spots around are occupied.

Now consider the one-step rolling process of $sPLA_2$ on the membrane surface.

First of all, assume that when the enzyme rolls to the active side (side A), the hydrolysation process may not happen, which means that it is possible that after roll to side A, the subjective enzyme may keep rolling to the other available sides.



In the graph above, the middle square represents the labeled enzyme (the alphabet in it shows the current side on the membrane), and the four spots on up, down, left, right denote the four spots that labeled enzyme will be able to roll to (the states that it may achieve in one-step rolling).

CASE 1 Enzyme is on side A

Since $sPLA_2$ lands is on side A and only one-step rolling events are being considered, it is easy to conclude that the transition probabilities from state A to the other states are

$$\mathcal{P}_{AT} = 0$$

$$\mathcal{P}_{\mathcal{AA}} + \mathcal{P}_{\mathcal{AS}} = 1$$

Labeled enzyme stops rolling if and only if all of the four spots connected are occupied by other enzymes. Therefore, the conditional transition probabilities given θ are

$$\mathcal{P}_{\mathcal{AA}|\theta=1} = 1$$

$$\mathcal{P}_{\mathcal{AS}|\theta=1} = 0$$

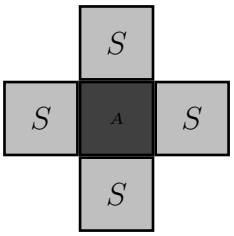
$$\mathcal{P}_{\mathcal{AA}|\theta=0} = \frac{1}{5}$$

$$\mathcal{P}_{\mathcal{AS}|\theta=0} = \frac{4}{5}$$

It is easy to conclude that the probabilities from A to A and A to S can be written

as

$$\begin{aligned}
\mathcal{P}_{AA} &= \mathcal{P}_{AA|\theta=1} \cdot p(\theta = 1) + \mathcal{P}_{AA|\theta=0} \cdot p(\theta = 0) \\
&= 1 \cdot p(\theta = 1) + \frac{1}{5} \cdot p(\theta = 0) \\
&= \frac{1}{5} \cdot p(\theta = 1) \\
\mathcal{P}_{AS} &= \mathcal{P}_{AS|\theta=1} \cdot p(\theta = 1) + \mathcal{P}_{AS|\theta=0} \cdot p(\theta = 0) \\
&= 0 \cdot p(\theta = 1) + \frac{4}{5} \cdot p(\theta = 0) \\
&= \frac{4}{5} \cdot p(\theta = 0)
\end{aligned}$$



CASE 2 Enzyme initially lands on side T

Since side T and side A are opposite in the cube, similarly, the following conclusions can be made:

$$\mathcal{P}_{TA} = 0$$

$$\mathcal{P}_{TT} + \mathcal{P}_{TS} = 1$$

Therefore the conditional transition probabilities given θ are

$$\mathcal{P}_{TT|\theta=1} = 1$$

$$\mathcal{P}_{TS|\theta=1} = 0$$

$$\mathcal{P}_{TT|\theta=0} = \frac{1}{5}$$

$$\mathcal{P}_{TS|\theta=0} = \frac{4}{5}$$

From the same method of Case 1, it can be conclude that

$$\begin{aligned}
\mathcal{P}_{TT} &= \mathcal{P}_{TT|\theta=1} \cdot p(\theta = 1) + \mathcal{P}_{TT|\theta=0} \cdot p(\theta = 0) \\
&= 1 \cdot p(\theta = 1) + \frac{1}{5} \cdot p(\theta = 0) \\
&= \frac{1}{5} \cdot p(\theta = 1) \\
\mathcal{P}_{TS} &= \mathcal{P}_{TS|\theta=1} \cdot p(\theta = 1) + \mathcal{P}_{TS|\theta=0} \cdot p(\theta = 0) \\
&= 0 \cdot p(\theta = 1) + \frac{4}{5} \cdot p(\theta = 0) \\
&= \frac{4}{5} \cdot p(\theta = 0)
\end{aligned}$$

CASE 3 Enzyme initially lands on side S

Due to the symmetry of side A and side T according to the position of side S, the transition probabilities from side S to A and T are identical. Thus, the rolling probability from side S to S is what requires to be considered.

(1) $\theta = 1$

When $\theta = 1$, all of the spots around the subjective enzyme are occupied by other enzymes, the rolling process of labeled enzyme is stopped. Thus, we have

$$\mathcal{P}_{SS|\theta=1} = 1$$

$$\mathcal{P}_{SA|\theta=1} = \mathcal{P}_{ST|\theta=1} = 0$$

$(2)\theta = 0$

When the spots around are all available to the subjective enzyme, the probabilities of rolling upward to side A, downward to side T, leftward, rightward to side S, and stay on side S are equal. Therefore,

$$\mathcal{P}_{SS|\theta=0} = \frac{3}{5}$$

$$\mathcal{P}_{SA|\theta=0} = \mathcal{P}_{ST|\theta=0} = \frac{1}{5}$$

From those three cases, it is easy to make the following conclusion:

- When $\theta = 0$,

$$\mathcal{P}_{AS} = \frac{4}{5}, \mathcal{P}_{AT} = 0, \mathcal{P}_{AA} = \frac{1}{5};$$

$$\mathcal{P}_{TS} = \frac{4}{5}, \mathcal{P}_{TT} = \frac{1}{5}, \mathcal{P}_{TA} = 0;$$

$$\mathcal{P}_{SS} = \frac{3}{5}, \mathcal{P}_{ST} = \frac{1}{5}, \mathcal{P}_{SA} = \frac{1}{5}$$

According to the definition of transition matrix in the beginning of this subsection, the transition matrix of the rolling process of enzyme $sPLA_2$ under the circumstance that $\theta = 0$ is as following:

$$\mathbb{P} = \begin{pmatrix} \frac{1}{5} & \frac{4}{5} & 0 \\ \frac{1}{5} & \frac{3}{5} & \frac{1}{5} \\ 0 & \frac{4}{5} & \frac{1}{5} \end{pmatrix} \tag{3}$$

where the rows of the above matrix represent the state of $sPLA_2$, and the columns represent the enzyme's state after one-step rolling. From the matrix above, the probability that enzyme $sPLA_2$ will end up with the active side (side A) after one-step rolling is

$$\begin{aligned} P(A) &= p(\text{land on A}) \cdot p_{AA} + p(\text{land on S}) \cdot_{SA} + p(\text{land on T}) \cdot p_{TA} \\ &= \frac{1}{6} \cdot \frac{1}{5} + \frac{1}{4} \cdot \frac{1}{5} + \frac{1}{6} \cdot 0 \\ &= \frac{1}{12} \end{aligned}$$

- When $\theta = 1$,

$$\mathcal{P}_{AS} = 0, \mathcal{P}_{AT} = 0, \mathcal{P}_{AA} = 1;$$

$$\mathcal{P}_{TS} = 0, \mathcal{P}_{TT} = 1, \mathcal{P}_{TA} = 0;$$

$$\mathcal{P}_{SS} = 1, \mathcal{P}_{ST} = 0, \mathcal{P}_{SA} = 0$$

The transition matrix is then

$$\mathbb{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

According to the matrix above,

$$\begin{aligned} P(A) &= p(\text{land on A}) \cdot p_{AA} + p(\text{land on S}) \cdot_{SA} + p(\text{land on T}) \cdot p_{TA} \\ &= \frac{1}{6} \cdot 1 + 0 \\ &= \frac{1}{6} \end{aligned}$$

The conditions of $\theta = 0$ and $\theta = 1$ are the two extreme situations. In reality, $\theta \in (0, 1)$ is a more common situation. If this is the case, a more complicated matrix will be concluded having θ as a variable in the matrix. This situation is not included in this paper.

In the next section, the classification of states will provide a long-term rolling probability from three states to the active state.

Assume that when the subjective enzyme rolls to side A, then the hydrolysation process will stop its rolling. Thus the probability that from side A to the other sides are all zero. Therefore, when $\theta = 0$, from the method that is introduced above, the transition matrix of the rolling process will be

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{5} & \frac{3}{5} & \frac{1}{5} \\ 0 & \frac{4}{5} & \frac{1}{5} \end{pmatrix} \tag{4}$$

3.2.2 Classification of States

Let $\mathbb{P}$ be the transition probability matrix of a Markov chain as defined in the previous sections. Let $p_j^{(n)}$ be the probability that the process is in state j after n transition. Denote by the row vector $\mathbf{p}^{(n)}$, the vector of probabilities $p_j^{(n)}$, $j \in S$. The **n**-step transition probabilities $p_{ij}^{(n)}$ are determined by the following theorem:

Theorem 2.1

$$\mathbf{P}^{(n)} = \mathbf{P}^n$$

and

$$\mathbf{P}^{(n)} = \mathbf{P}^{(0)}\mathbf{P}^n.$$

In order to prove this theorem, we need to introduce the Chapman-Kolmogorov equation as following.

When the stochastic process has a discrete state space and a discrete parameter space, for $n > n_1 > n_2 > ... > n_k$ and $n, ..., n_k$ belonging to the parameter space,

$$\begin{aligned} P(X_n = j \mid X_{n_1} &= i_1, ..., X_{n_k} = i_k) \\ &= P(X_n = j \mid X_{n_1} = i_1) \\ &= P_{i_1 j}^{(n_1, n)} \end{aligned}$$

Using this property, from $m < r < n$, it is easy to conclude the Chapman-Kolmogorov equation

$$\begin{aligned} P_{ij}^{(m,n)} &= P(X_n = j \mid X_m = i) \\ &= \sum P(X_n = j \mid X_r = k) P(X_r = k \mid X_m = i) \\ &= \sum P_{ik}^{(m,r)} P_{kj}^{(r,n)} \end{aligned}$$

The following is the proof of Theorem 2.1.

Proof From the Chapman-Kolmogorov equation, we have

$$P_{ij}^{(r+s)} = \sum P_{ik}^{(r)} P_{kj}^{(s)} \text{ for given r and s.}$$

Set $r = 1$, $s = 1$, the above equation will be

$$P_{ij}^{(2)} = \sum P_{ik} P_{kj}$$

Clearly, $P_{ij}^{(2)}$ is the (i, j)th element of the matrix product $P \cdot P = P^2$. Based on this result, assume that

$$P^{(r)} = P^r, r = 1, 2, ..., n-1$$

Setting r = n-1, s = 1, then

$$P_{ij}^{(n)} = \sum P_{ik}^{n-1} P_{kj}$$

which again can be seen as the (i, j)th element of the matrix product $P^{n-1} \cdot P = P^n$, which proves the first result of the theorem. The second result is obtained by noting that

$$P(X_n = j) = \sum P(X_n = j \mid X_0 = i)P(X_0 = i)$$

QED

From this theorem it is clear that when the size of the state space is small, and the n-step transition probabilities can easily be obtained by simple matrix multiplication. For large state spaces, efficient methods for the calculation of P^n are needed.

For example, from the last section, when $\theta = 0$, the transition matrix of the rolling process of enzyme $sPLA_2$ is

$$\mathbb{P} = \begin{pmatrix} \frac{1}{5} & \frac{4}{5} & 0 \\ \frac{1}{5} & \frac{3}{5} & \frac{1}{5} \\ 0 & \frac{4}{5} & \frac{1}{5} \end{pmatrix}$$

Since it is proved that the powers of transition matrix give us interesting information about the process as it evolves. The state of the subjective enzyme after a large number of steps is very interesting. MATLAB is applied to compute the successive powers of $\mathbb{P}$. It is obvious that after seven steps our state predictions are, to four-decimal-place accuracy, independent of the original state. The probabilities for the three types of sides, A, S and T are 0.1667, 0.6667 and 0.1667 no matter where the chain started. This is an example of a type of Markov chain called a *regular* Markov chain. For this type of chain, it is true that long-range predictions are independent of the starting state.

Definition A Markov chain is called a *regular* chain if some power of the transition matrix has only positive elements.

In other words, for some n, it is possible to go from any state to any state in exactly n steps. Now with the theorem of n-step transition probabilities and the definition of a regular Markov Chain, it is easier to define the classes of states.

There are two important theorems relating to regular chains. The following theorems and proofs can be easily found in [5].

Theorem 2.2 (Fundamental Limit Theorem for Regular Chains) Let $\mathbf{P}$ be the transition matrix for a regular chain. Then, as $n \to \infty$, the powers $\mathbf{P}^n$ approach a limiting matrix $\mathbf{W}$

$$\lim \mathbf{P}^n = \mathbf{W},$$

where $\mathbf{W}$ is a matrix with all rows the same vector $\mathbf{w}$. The vector $\mathbf{w}$ is a strictly positive probability vector (i.e., the components are all positive and they sum to one).

The proof of this theorem will be provided after the introduction of a few important tools and definitions.

Theorem 2.3 Let $\mathbf{P}$ be a regular transition matrix, let

$$\mathbf{W} = \lim \mathbf{P}^n,$$

let $\mathbf{w}$ be the common row of $\mathbf{W}$, and let $\mathbf{c}$ be the column vector all of whose components are 1. Then

(a) $\mathbf{wP} = \mathbf{w}$, and any row vector $\mathbf{v}$ such that $\mathbf{vP} = \mathbf{v}$ is a constant multiple of $\mathbf{w}$.

(b) $\mathbf{Pc} = \mathbf{c}$, and any column vector $\mathbf{x}$ such that $\mathbf{Px} = \mathbf{x}$ is a multiple of $\mathbf{c}$.

Proof. To prove part (a), from theorem 2.2,

$$\mathbf{P}^n \to \mathbf{W}.$$

Thus,

$$\mathbf{P}^{n+1} \to \mathbf{P}^n \cdot \mathbf{P} \to \mathbf{WP}.$$

But $\mathbf{P}^{n+1} \to \mathbf{W}$, and so $\mathbf{W} = \mathbf{WP}$, and $\mathbf{w} = \mathbf{wP}$.

Let $\mathbf{v}$ be any vector with $\mathbf{vP} = \mathbf{v}$. Then $\mathbf{v} = \mathbf{vP}^n$, and passing to the limit, $\mathbf{v} = \mathbf{vW}$. Let r be the sum of the components of $\mathbf{v}$. Then it is easily checked that $\mathbf{vW} = r\mathbf{W}$. So, $\mathbf{v} = r\mathbf{w}$.

To prove part (b), assume that $\mathbf{x} = \mathbf{Px}$. Then $\mathbf{x} = \mathbf{P}^n\mathbf{x}$, and again passing to the limit, $\mathbf{x} = \mathbf{Wx}$. Since all rows of $\mathbf{W}$ are the same, the components of $\mathbf{Wx}$ are all equal, so $\mathbf{x}$ is a multiple of $\mathbf{c}$.

QED

Note that an immediate consequence of Theorem 2.3 is the fact that there is only one probability vector $\mathbf{v}$ such that $\mathbf{vP} = \mathbf{v}$.

Definition A row vector $\mathbf{w}$ with the property $\mathbf{wP} = \mathbf{w}$ is called a *fixed row vector* for $\mathbf{P}$. Similarly, a column vector $\mathbf{w}$ such that $\mathbf{Px} = \mathbf{x}$ is called a *fixed column vector* for $\mathbf{P}$.

Here are some common methods for calculating the fixed row vector **w** for a regular Markov chain.

Take matrix $\mathbb{P}$ in (2) as an example. By the Fundamental Limit Theorem, there is a limiting vector **w** of matrix **P** from the fact that

$$w_1 + w_2 + w_3 = 1$$

and

$$(w_1 w_2 w_3) \begin{pmatrix} \frac{1}{5} & \frac{4}{5} & 0 \\ \frac{1}{5} & \frac{3}{5} & \frac{1}{5} \\ 0 & \frac{4}{5} & \frac{1}{5} \end{pmatrix} = (w_1 w_2 w_3)$$

under the assumption that the rolling process is not stopped when the enzyme rolls to side A.

These relations lead to the following four equations in three unknowns:

$$\begin{aligned} w_1 + w_2 + w_3 &= 1, \\ \frac{1}{5}w_1 + \frac{1}{5}w_2 + 0w_3 &= w_1, \\ \frac{4}{5}w_1 + \frac{3}{5}w_2 + \frac{4}{5}w_3 &= w_2, \\ 0w_1 + \frac{1}{5}w_2 + \frac{1}{5}w_3 &= w_3. \end{aligned}$$

The theorem guarantees that these equations have a unique solution. If the equations are solved, the solution is

$$\mathbf{w} = \begin{pmatrix} 0.1667 & 0.6667 & 0.1667 \end{pmatrix}$$

in agreement with that predicted from $\mathbb{P}^{15}$, given above.

To calculate the fixed vector, we can assume that the value at a particular state, say state A, is 1, and then use all but one of the linear equations from $\mathbf{wP} = \mathbf{w}$. This set of equations will have a unique solution and $\mathbf{w}$ will be obtained from the solution by dividing each of its entries by their sum to give the probability vector $\mathbf{w}$. Matrix $\mathcal{P}$ will be applied to illustrate this idea.

Let $w_1 = 1$, and then solve the first and second linear equations from $\mathbf{wP} = \mathbf{w}$. We have

$$\begin{aligned} \frac{1}{5}w_1 + \frac{1}{5}w_2 + 0w_3 &= w_1, \\ \frac{4}{5}w_1 + \frac{3}{5}w_2 + \frac{4}{5}w_3 &= w_2. \end{aligned}$$

From the equations above, it is easy to obtain

$$\begin{pmatrix} w_1 & w_2 & w_3 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 1 \end{pmatrix}.$$

Now divide this vector by the sum of the components, to obtain the final answer:

$$\mathbf{w} = \begin{pmatrix} 0.1667 & 0.6667 & 0.1667 \end{pmatrix}.$$

This method can be easily programmed to run on a computer.

Theorem 2.4 Let P be the transition matrix for a regular chain and $\mathbf{v}$ an arbi-

trary probability vector. Then

$$\lim \mathbf{v}\mathbf{P}^n = \mathbf{w},$$

where $\mathbf{w}$ is the unique fixed probability vector for P.

Proof By Theorem 2.2,

$$\lim \mathbf{P}^n = \mathbf{W}.$$

Hence,

$$\lim \mathbf{v}\mathbf{P}^n = \mathbf{v}\mathbf{W}.$$

But the entries in $\mathbf{v}$ sum to 1, and each row of $\mathbf{W}$ equals $\mathbf{w}$. From these statements, it is easy to check that

$$\mathbf{v}\mathbf{W} = \mathbf{w}.$$

QED

If we start a Markov chain with initial probabilities given by $\mathbf{v}$, then the probability vector $\mathbf{v}P^n$ gives the probabilities of being in the various states after n steps.

Apply Theorem 2.4 to the rolling process of $sPLA_2$. From the assumptions that are made in the second chapter, since the shape of the enzyme is hypothesized to be a cube with one side of A, four sides of S, and one side of T, the initial probabilities that the enzyme will land on Side A, S and T will be

$$\mathbf{v} = (\frac{1}{6}, \frac{4}{6}, \frac{1}{6}).$$

Thus, the probabilities of being in state A (on side A), state S and state T after the fifteenth step will be

$$\mathbf{v}P^{15} = \begin{pmatrix} 1/6 & 4/6 & 1/6 \end{pmatrix} \begin{pmatrix} 0.1667 & 0.6667 & 0.1667 \\ 0.1667 & 0.6667 & 0.1667 \\ 0.1667 & 0.6667 & 0.1667 \end{pmatrix} = (0.16670.66670.1667)$$

which turns out to be the same result as was concluded from the definition of the fixed row vector.

After analyzing the matrix $\mathbf{P}$ in (3), it becomes necessary to further examine matrix

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{5} & \frac{3}{5} & \frac{1}{5} \\ 0 & \frac{4}{5} & \frac{1}{5} \end{pmatrix}$$

which was conclude under the assumption that the rolling process stops when the enzyme reach at state A.

Definition State j is said to be accessible from state i if j can be reaches from i in a finite number of steps. If two states i and j are accessible to each other, then they are said to *communicate*. Probabilistically these definitions imply:

• $i \to j$ (j accessible from i) if for some $n \geq 0$, $P_{ij}^{(n)} > 0$

• $j \to i$ (i accessible from j) if for some $n \geq 0$, $P_{ji}^{(n)} > 0$

• $i \leftrightarrow j$ (i and j communicate) if for some $n \geq 0$ and $m \geq 0$, such that $P_{ij}^{(n)} > 0$ and $P_{ij}^{(n)} > 0$

As a consequence of the definition, the following properties of this communication relation:

1. *Reflexivity* . $i \leftrightarrow i$ for

$$P_{ij}^{(0)} = \begin{cases} 1 \text{ if i = j} \\ 0 \text{ if i} \neq \text{j}. \end{cases}$$

2. *Summetry* . If $i \leftrightarrow j$, then $j \leftrightarrow i$.

3. *Transitivity.* If $i \leftrightarrow j$ and $j \leftrightarrow k$, then $i \leftrightarrow k$.

Incidentally, it may be mentioned that the properties of communicate states define an equivalence relation and hence the communication relation is an equivalence relation. The set of all states of Markov chain that communicate (with each other) can therefore be grouped into a single equivalence class. A Markov chain may have more than one such equivalence class. If there are more than one, then it is not possible to have communicating states in different equivalence classes. However, it is possible to have states in one class that are accessible for another class.

Definition A state i is said to be *recurrent* if and only if, starting from state i, eventual return to this state is certain.

Take matrix **P** of (2) as an example. State A is a recurrence state, since from the assumption that is made in the last section, as soon as the subjective enzyme rolls to side A, the hydrolysation process will being initiated, and the rolling process will be stopped immediately; thus it is impossible for the subjective enzyme to roll to other

sides.

Definition A state i is said to be *transient* if and only if, starting from state i, there is a positive probability that the process may not eventually return to this state.

In matrix **P** in (3), state T is an example of a transient state. It is because there is a probability of $\frac{4}{5}$ that the subject enzyme will roll from state T to state S, and a probability of $\frac{1}{5}$ from state S to state A. Since as soon as it rolls to state A, there is no possibility that it will roll back to state T, the probability that the process may not eventually return to state T is

$$P = \frac{4}{5} \cdot \frac{1}{5} = \frac{4}{25} > 0.$$

From the similar method, it is easy to see that state S is also a transient state in matrix P.

There are a lot of questions concerning these two different type of states: Given that the process is in a transient state i initially (for example, side S), what is the average number of visits it makes to anther transient state j (for example, side T) before it eventually enters any one of the recurrent states? What is the variance of the number of visits to j from i? In order to fully understand matrix P, the fundamental matrix $M = (I - Q)^{-1}$ will be applied. This matrix plays a useful role in the determination of the means and variance mentioned above, and it is a very important tool to analyze the behavior of the Markov chain in the presence of the transient states.

Let the $m-$state Markov chain consist of r recurrent states and $(m - r)$ transient states with the latter belonging to a single equivalence class. Let T be the set of these

transient states and T^c the set of recurrent states. The transition probability matrix Pcan now be put in the form

$$\mathbf{P} = \begin{pmatrix} P_1 & 0 \\ R & Q \end{pmatrix}$$

where P_1 is an $r \times r$ submatrix with transition probabilities among the recurrent states for its elements; Q is an $(m-r) \times (m-r)$ substochastic matrix (with at least one row sum less than 1) with probabilities of transition only among the transient states for its elements; and R is an $(m-r) \times r$ submatix whose elements are the probabilities of the one-step transition from $(m-r)$ transient states to the r recurrent states.

Let $N_{ij}(i, j \in T)$ be the random variable denoting the number of times the process visits j before it eventually enters a recurrent state, having initially started from state i. Let $(\mu_{ij})_{i \times j} = N_{ij}$

Theorem 2.5 For $i, j \in T$

$$(\mu_{ij})_{i \times j} = M.$$

Proof Initially, the Markov chain is in state $i \in T$. If in one step it enters a recurrent state (with probability $\sum P_{ik}$), the number of visits to j is zero unless $j = i$. If δ_{ij} is the Kronecker δ function such that $\delta_{ij} = 1$ if $j = i$ and 0 if $i \neq j$, we can write $N_{ij} = \delta_{ij}$ with probability $\sum P_{ik}$. On the other hand, suppose the Markov chain moves to a state $k \in T$ at the first step (with probability P_{ik}). From that position onward, the number of the visits to j is N_{kj}. However, if $i = j$, the total number of visits to j would be $N_{kj} + \delta_{ij}$. Thus we have

$$N_{ij} = \begin{cases} \delta_{ij}, & \text{with probability} \sum P_{ik} \\ N_{kj} + \delta_{ij}, & \text{with probability} P_{ik} k \in T \end{cases}$$

Taking expectations, we get

$$E(N_{ij} = \sum P_{ik}\delta_{ij} + \sum P_{ik}E(N_{kj} + \delta_{ij}))$$

which gives

$$\mu_{ij} = \delta_{ij} + \sum P_{ik}\mu_{kj}$$

Using all the elements of the matrix

$$(\mu_{ij})_{i\times j} = I + Q(\mu_{ij})_{i\times j}$$

Hence

$$(I - Q)^{-1} = M$$

QED

Related to the matrix M, define the matrices that follow. Let $M = (\mu_{ij})_{i\times j}$ $(i, j \in T$ and $i, j = r+1, r+2, ..., m)$;

$$M_D = \begin{pmatrix} \mu_{r+1,r+1} & & 0 \\ & \mu_{r+2,r+2} & \\ & ... & \\ 0 & & \mu_{m,m} \end{pmatrix}$$

and

$$M_2 = \begin{pmatrix} \mu^2_{r+1,r+1} & \mu^2_{r+1,r+2} & \cdots & \mu^2_{r+1,m} \\ \mu^2_{r+2,r+1} & \mu^2_{r+2,r+2} & \cdots & \mu^2_{r+2,m} \\ & \cdots & \cdots & \\ \mu^2_{m,r+1} & \mu^2_{m,r+2} & \cdots & \mu^2_{m,m} \end{pmatrix}$$

$$M_\rho = (\sum_{j \in T} \mu_{ij})$$

$$M_{\rho^2} = (\sum_{j \in T} \mu_{ij})^2$$

Theorem 2.6

$$(E(N_i)) = M_\rho, i \in T$$

$$(E(N_i^2)) = M_{\rho^2}, i \in T$$

where $E(N_i)$ is the expected number steps from the transient states to the recurrent states.

Let $\sigma^2_{ij} = V(N_{ij})$.

Theorem 2.7

$$\sigma^2_{ij} = M(2M_D - I) - M_2, i, j \in T$$

where σ is the variance from the transient states to the recurrent states.

Therefore, in order to figure out the expected number and variance steps that an enzyme will roll to side A eventually, the above two theorems can be applied to the

matrix that was obtained when $\theta = 0$ in (3)

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{5} & \frac{3}{5} & \frac{1}{5} \\ 0 & \frac{4}{5} & \frac{1}{5} \end{pmatrix}$$

so that

$$Q = \begin{pmatrix} \frac{3}{5} & \frac{1}{5} \\ \frac{4}{5} & \frac{1}{5} \end{pmatrix}$$

and

$$M = (I - Q)^{-1} = \begin{pmatrix} 5 & 1.25 \\ 5 & 2.5 \end{pmatrix}$$

From M we get

$$M_D = \begin{pmatrix} 5 & 0 \\ 0 & 2.5 \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 25 & 1.56 \\ 25 & 6.25 \end{pmatrix}$$

$$M_\rho = \begin{pmatrix} 6.25 \\ 7.5 \end{pmatrix}$$

$$M_{\rho^2} = \begin{pmatrix} 39.0625 \\ 56.25 \end{pmatrix}$$

Using Theorem 2.6.3, we get

$$\sigma_{ij}^2 = M(2M_D - I) - M_2 = M_D = \begin{pmatrix} 20 & 3.44 \\ 20 & 3.75 \end{pmatrix}.$$

The following conclusions can be drawn from the above results: Suppose an enzyme is at side S. Before it is finally rolls to side A, the expected number of steps that it will be on side S is 5, with a variance of 20; the expected number of steps that it will be on side T is 1.25, with variance 3.44. Further, the total time before the enzyme rolls to side A has an expected value of 6.25 steps, with variance 39.0625. Similar interpretations can be given to the results regarding those enzyme that start at side T: the expected time that it will be on side S is also 5, with a variance 20; the expected number of steps that it will be on side T is 2.5, with variance 3.75; the total time before it rolls to side A is 7.5 steps, with variance 56.25.

3.3 Lattice Gas Automata

3.3.1 Lattice Gas Automata and Monte Carlo Method

Suppose the joint distribution between two variables X and Y is $f(X, Y)$, but f has a complicated mathematical form. If we wish to determine the expected value of a function $g(X, Y)$ $(E[g(X, Y)])$, one way to accomplish this is through *Monte Carlo simulation*. Such a simulation involves generating independent realization from the density f and for each such realization calculating $Z = g(X, Y)$. Thus, after n realizations of f are simulated, we have the values $Z_1, Z_2, ..., Z_n,$, where $Z_i = g(X_i, Y_i)$ and (X_i, Y_i) is the ith simulated observation from f. A law of large numbers will then allow us to conclude that

$$\lim_{n\to\infty} \sum_{i=1\to n} Z_i/n = E[g(X, Y)]$$

Although Monte Carlo methods are used in a diverse number of ways, in the context of molecular computations there are five types most commonly encountered. In this paper, a simulation Monte Carlo or SMC will be applied. This Monte Carlo method is a series of stochastic algorithms that are used to generate initial conditions to actually simulate processes using scaling arguments to establish time scales or by introducing stochastic effects into molecular dynamics.

Lattice gas automata, or LGA is a particular type of simulation that is used for the viscous fluid flow. LGA research is a highly developed subculture of general cellular automation research. This tool has been widely used in analyzing the dynamics of reactions in a variety of conditions. A microscopic image of the detailed reaction can be obtained through the application of theoretical framework. This image is very useful for helping us gain a more detailed understanding of the dynamics of biochemistry reactions. Recently, it has been used to model the Michaelis-Menten mechanism on a two-dimensional grid with a cyclic boundary. In this section a Monte Carlo algorithm combined with LGA simulation will be applied for the hydrolysation process of enzyme $sPLA_2$ to (i) understand the dynamics of the hydrolysation of rolling process, (ii) establish a modification of the rolling ordinary differential model and (iii) verify the other results that were obtained in the second chapter.

3.4 LGA Model using Monte Carlo algorithm

Following [1], we implement a lattice gas automata model in C++ using a Monte Carlo algorithm on a two-dimensional square lattice with cyclic boundary conditions. In the model, each molecule is mobile on the lattice through diffusion, modeled by independent nearest-neighbor random walks of the individual molecule.

In the LGA model, time is split into discrete steps and at each step enzyme are selected at random. After that, randomly choose a grid site among the nearest positions of the subjective enzyme. If the chosen spot is empty, move the enzyme into the vacancy; otherwise, keep the enzyme in its original position. In this method, enzyme move through the volume via two-dimensional random walks known as blind ant process [9].

Let the size of each sub-square of the grid be as the same as the size of an enzyme; thus, after one step rolling, the enzyme will reach another site. According to the initial condition of the subjective enzyme, a parameter, which is devrived by Section 2 will be used to determine which side the enzyme will move onto. When an enzyme moves to a site with a phospholipid, and the enzyme is on its active site, then a hydrolysation process may happen. After the reaction, the lipid will be removed, the enzyme will be released from the membrane to the solution and one product P will be produced. The coordinates of the position of every enzyme and occupancy status of each lattice site are stored and used for analysis. At any moment of the simulation, one given lattice site cannot be occupied by more than one molecule.

Let γ be the number of product P and the rolling and reacting probabilities k_1, k_2 and k_{cat} are defined as following:

- k_1 is the first entry of the fixed probability vector $\mathbf{w}$ in the previous chapter.
- k_2 is the second entry of the fixed probability vector $\mathbf{w}$ in the previous chapter.
- k_{cat} is the reaction coefficient that is applied in the ODE model.

At the beginning of the LGA model, the enzyme and the lipid are placed on the lattice by randomly choosing the co-ordinates for each of them. At each Monte Carlo simulation, a *subjective* enzyme is chosen at random and the rolling of the enzyme

is according to the following rules:

1. Randomly choose one from all of the occupied spots. If this spot is occupied, keep the subjective enzyme on its original spot.

2. Otherwise move the subjective enzyme into the vacancy according to the following rules:

• If the enzyme is on side S, randomly generate a number between 0 and 1. If this number is smaller than probability k_1 , move the enzyme to the vacancy and roll it to side A; if the random number is between k_1 and k_2, move the enzyme to the vacancy and roll it to side S; otherwise roll it to side T. This is the end of the first step rolling process.

• If the enzyme is originally on its top side or active side, replace the enzyme from its original spot, move it to the vacancy and roll it to side S.

After the rolling process, the hydrolysation occurs according to the following rules:

1. If the enzyme on the grid spot is not on its active side or no available lipid on that spot, no reaction happens.

2. Otherwise generate a random number from 0 to 1. Compare this number with reaction coefficient k_{cat}. If the number is smaller than k_{cat}, reaction happens, which means the lipid and the enzyme will be removed from the grid spot and set $\gamma = \gamma + 1$. Otherwise, keep the enzyme and the lipid on the membrane.

The reaction is not the end of the simulation. Since enzyme's density in the solution is very high [2], as soon as the reaction happens, an enzyme in the solution will land on an available spot; therefore, the number of enzyme on the membrane is always constant. Now we need to figure out this constant. The dimension of $sPLA_2$ is $45\mathring{A} \times 30\mathring{A} \times 20\mathring{A}$ ($1\mathring{A} = 10^{-10}m$)[6], and since it is assumed that the geometry shape of an enzyme is a cube, a size of $45\mathring{A} \times 45\mathring{A} \times 45\mathring{A}$ is what we applied in our model. Thus the area of our grid is

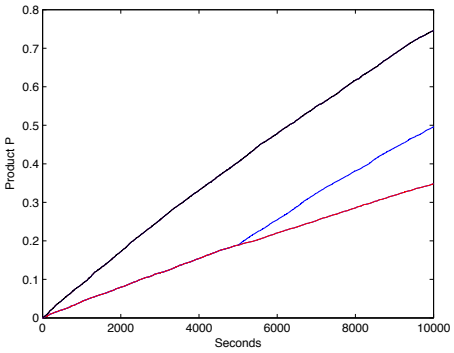


Figure 7: Low Density of Enzyme with a High Level of Lipid: This plot shows the changes of product P as the increase of length of time. The red curve represents the experiment when 306 of enzyme are added as an initial value; the blue line represents the reaction when 306 enzyme are added in the beginning and then the same amount is added in the middle of the reaction; the black line shows the changes when twice amount of enzyme are added in the beginning of the experiment. Fig. 7-10 all follows this label notation. Also, these four graphs all come from the C++ code in the Appendix 5.2.

$$(45\mathring{A} \times 99)^2 = 1.98 \times 10^{-13} m^2$$

The average surface area of a single blood cell is about $129.9\mu M^2$ ($1\mu M = 10^{-6}m$)[4]. The average number of enzyme that absorbed in all Bell's experiments was $2.0 \pm 0.3 \times 10^5$. Therefore, the number of enzyme that we should keep on our grid is

$$n_{total} = \frac{2 \times 10^5 \times 1.98 \times 10^{13}}{129 \times 10^{-12}} \approx 306$$

Figure 7 to 10 are the results of this LGA Monte Carlo simulation. Those graphs show the changes of product P within the first 10^4 seconds. Figure 7 is obtained when the number of enzyme is 306 (a low density) and the number of available lipids is 30 percent of the total number of grid spots. The red curve represents the experiment when 306 of enzyme are added as an initial value; the blue line represents the reaction when 306 enzyme is added in the beginning and then the same amount is added in the middle of the reaction; the black line shows the changes when twice amount of

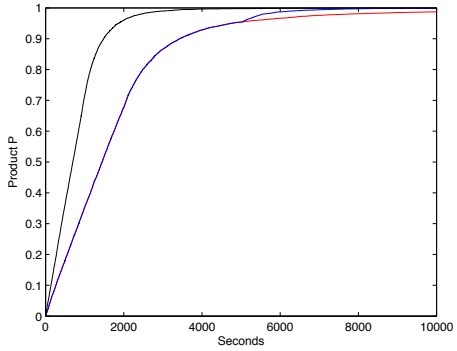


Figure 8: High Density of Enzyme with a High Level of Lipid: This plot shows the changes of product P as the increase of length of time with an initially 3000 unit of enzyme and a relatively large amount of lipids (assume 30% of the total grid spots have lipids available). The red curve shows the experiment when 3000 enzyme are added as initial value of enzyme; blue line shows the reaction when 3000 are added in the beginning and another 3000 is added in the middle; black curve shows the changes when 6000 initial enzyme are added in the beginning of the experiment.

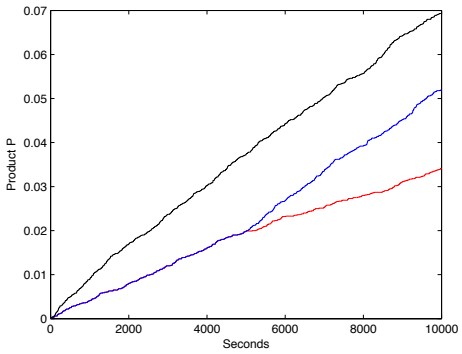


Figure 9: Low Density of Enzyme with a low Level of Lipid: This plot shows the changes of product P as the increase of length of time with an initially low amount (306) of enzyme and small amount of lipids (assume only 3% of the total grid spots have available lipids).

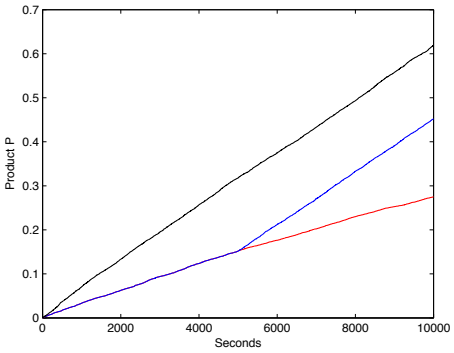


Figure 10: High Density of Enzyme with a low Level of Lipid: This plot shows the changes of product P as the increase of length of time with an initially large amount of enzyme (3000) and limited lipids (3%).

enzyme is added in the beginning of the experiment.

From this graph, it is easy to see that, when the initial amount of enzyme is twice as much (black curve), the speed of the reaction will be faster comparing with the red line. Also, when an extra amount of enzyme is added in the middle of the experiment, there is a very evident increase in the rate of production of P. However, when the same amount of enzyme is added, no obvious increase of product P is observed [17]. One possible explanation of this result is that a crowding on the membrane surface may stopped the enzyme from rolling; however, the amount of enzyme in our simulation is too small to establish crowding (only 0.3% of sites are occupied); thus, when more enzyme is added in the middle, more enzyme are absorbed. In order to verify the relationship between crowding and the amount of product P, we need to compare Figure 8 and 7.

Figure 8 shows the changes of product P with a high density of enzyme on the membrane. Initially, 3000 enzyme are put on the membrane. Apparently, the reaction speed is much faster than Figure 5, which gives us the same result that we obtained above: more enzyme will lead a faster reaction speed. In Figure 8, the red line shows

that when an extra amount of enzyme is added in the middle of the experiment, the changes in the amount of product is not as much as in Figure 7, but it is still quite obvious. It shows that the crowding on the membrane surface is not the main reason to the halt of the reaction. When there is a crowding on the membrane, the number of reactions is still increased a lot. Therefore, there must be some other reasons that cause the halt of the reaction [17].

Figure 9 is obtained when the available lipids are really limited (3% of the total number of grid spots) and the initial enzyme amount is relatively low. From Figure 9, the total amount of product in the first 10^4 seconds is about 10% of Figure 7; also, the reaction is slowed down–the time to achieve a steady state is much longer than a higher density lipid membrane. An interesting observation is when an extra amount of enzyme is added on the membrane in the middle of the reaction (blue line), the amount of product that is increased is also smaller than Figure 7. This may due to the fact that the limitation of available lipids slows down the reaction speed, and decrease the total amount of product that we may obtain.

Figure 10 indicates the changes of product under a high enzyme density but low lipid level. Comparing it with Figure 8, it is easy to see that the reaction is slowed down because of the low density of lipids; also, when an extra amount of enzyme is added in the middle, quite an observable increase of the amount of product can be viewed.

Therefore, this LGA Monte Carlo simulation shows us that the density of enzyme determines the speed of the reaction. Under the same level of lipids, more enzyme will have more lipids hydrolyzed, which will end up with a relatively high amount of product for each time step; therefore the reaction speed will be increased, and the

time that the reaction needs to go to a steady state will be shorten. Under a high density enzyme circumstance, extra enzyme will still result an obvious increase in the amount of product. Even though the membrane is already packed with enzyme, and the density of enzyme in the solution is very high, there is still a relatively large amount of extra product will be observed in the simulation. Thus, more enzyme in the solution is actually helpful with the hydrolyzing, and the result in Bell's experiment must come from other reasons than the halt of rolling process. The lipid density is also very important in determining the reaction speed and product amount. When the amount of available lipid on the membrane is limited, the enzyme on the membrane will need to roll more steps to reach an available lipid, which is why the reaction speed is slowed down.

4 Conclusions

In the ODE model, we concluded that the hydrolyzing speed of of enzyme $sPLA_2$ will increase with a high density of initial amount of enzyme. Also, more product will be observed when an extra amount of enzyme is added in the middle of the experiment. Applying Transition Matrix, we calculated the initial amounts of enzyme which land on side A, side S and side T when it is bind on the blood cell membrane, which cooperate with the amount that can be observed in the ODE graphs (Figure 2 and 3).

After that, we use LGA Monte Carlo simulation to establish a rolling grid to show a microscopic image of the rolling process of enzyme. This LGA model indicates when the density of the enzyme on the membrane is relatively low, more product will be observed when an extra amount of enzyme is added in the middle of the experiment. This result comes from the fact that when the enzyme density is low, there is a lot

of area with available lipid that the enzyme is not able to reach; thus when a large amount of enzyme is injected, a lot of them will be able to land on those areas to hydrolyze the available lipid. Even though no extra product is observed in Bell's experiment, in our simulation when there is crowding on the membrane there still is an increase in the amount of product due to the extra enzyme. Thus, the rolling process is not the main reason.

Therefore, a lot of other conditions may be considered as the main reason. First of all, our model assumed that the enzyme can roll freely on the cell membrane unless another enzyme is on its rolling direction. However, is it possible that an enzyme will be trapped in some spots and the rolling process will be stopped? If it is possible, what is the percentage of this kind of trapping spots on the membrane surface? How are these spots distributed on the membrane? How large is its influence to the hydrolyzing process? Or is it possible that enzyme can only go through binding spots to binding spots? How do these binding spots distribute? Will its distribution influence the reaction? Also, are there any relationship between the distribution of binding spots and the distribution of lipids? When an enzyme is rolling through the binding spots, what is the availability of lipids that the enzyme may hydrolyze?

There are few new methods and tools that a number of authors are applying to the reaction kinetics to study the macromolecular crowding. For example, Zipf-Mandelbrot distribution and the probability density function of Gillespie. How do these tools help us learn the rolling process and what are the advantage and disadvantage of those models? All of these questions should be considered in further research.

[1] Atsumi, G., M. Murakami, M. Tajima, S. Shimbara, N. Hara, and I. Kudo. 1997. The perturbed menbrane of cells undergoing apoptosis is susceptible to type II secretory phospholipase A2 to liberate arachidonic acid. Biochim. Biophys. Acta 1349:43-54

[2] Berg, B., 2004. Markov Chain Monte Carlo simulations and their statistical analysis. World Scientific.

[3] Berry, H., 2002. Monte Carlo simulations of enzyme reactions in two dimensions: fractal kinetics and spatial segregation. Biophys. J. 83, 1891-1901.

[4] Best, K., A. Ohran, A. Hawes, T. L. Hazlett, E. Gratton, A. M. Judd, and J. D. Bell, 2002. Relationship between erythrocyte membrane phase properties and susceptibility to secretory phospholipase A_2. Biochemistry. 41, 13982-13988.

[5] Bhat, U., Miller, G., 2002. Elements of Applied Stochastic Processes. A John Wiley & Sons, Inc., Hoboken, New Jersey.

[6] Colman, A., 1995. Game theory and its application in the social and biological science. Garkabd Science. 12-20.

[7] Fung, Y.C., 1993. Biomechanice-Mechanical properties of living tissues. Springer-Verlag, New York.

[8] Koduri, R. S., S. F. Baker, Y. Snitko, S.K.Han, W. Cho, D.C. Wilton, and M. H. Gelb. 1998. Action of han group IIa secreted phospholipase A2 on cell membranes. Vesicle but not heparinoid binding determines rate of fatty acid release by

exogenously added enzyme. J. Biol. Chem. 237: 32142-32153

[9] Majid, I., Ben Avraham, D., Havlin, S., Stanley, H. E., 1984. Exact-enumeration approach to random walks on percolations clusters in two dimensions. Phys. Rev. B 30, 1626-1628.

[10] Menashe, M., D. Lichtenberg, C. Gutierrez-Merino, and R. L. Biltonen. 1981. Relationship between the activity of pancreatic phospholipase A2 and the physical state of the phospholipid substrate. J. Biol. Chem. 256:4541-4543.

[11] Michaelis, L., and Menten, M. L. 1913. Die kinetik der invertinwirkung. Biochem. J., 19, 339-339.

[12] Misra, J. C., 2006. Biomathematics: Modeling and Simulation. World Scientific.

[13] Goel, N. S., Richter-dyn, N., 1974. Stochastic Models in Biology. Academic Press,Inc., New York.

[14] R. K. Arni and R. J. Ward, 1996. Phospholipase A_2 - A Structural Review. Elsevier Science Ltd. 34, 827-841.

[15] R. M. May, 2004. Uses and Abuses of Mathematics in Biology. Science.

[16] Schnell, S., and T. E. Turner, 2004. Reaction kinetics in intracellular environments with macromolecular crowding: simulations and rate laws. Elsevier Science 85, 235-260.

[17] Smith, S. K., A. R. Frarnbach, F. M. Harris, A. C. Hawes, L. R. Jackson, A. M. Judd, R. S. Vest, S. Sanchez, and J. D. Bell. 2001. Mechanisms by which

intracellular calcium induces susceptibility to secretory phospholipase A2 in human erythrocytes. J. Biol. Chem. 276: 22732-22741.

[18] T. M. Apostol, 1969. Calculus. Vol II. John Wiley & Sons, Inc., New Jersey.

[19] T. Ohba, S. Tsuchiya, T. Kumeta and K. Ohki. 2005. Microscopic imaging of phase separation in Giant liposome by using laurdan at video rate. Japanese Journal of Applied Physics. 44:8733-8738.

[20] Todhunter, I., 1865. History of the mathematical theory of probability from the time of Pascal to that of Laplace. Macmillan and Co., Cambridge and London.

[21] Vest, R., Gonzales, L., Permann, S., Spencer, E., Hansen, L., Judd, A., and J. D. Bell, 2004. Divalent Cations Increase Lipid Order in Erythrocytes and Susceptibility to Secretory Phospholipase A_2. Biophys. J. 86, 2251-2260.

[22] Wilson, H., Waldrip, J., Nielson, K., Judd, A., Han, S., Cho, W., Sims, P., J. D. Bell, 1999. Mechanisms by which elevated intracellular Calcium induces S49 cell membrane to become susceptible to the action of Secretory Phospholipase A_2. Biochemistry. 274, 11494-11504.

5 Appendix

5.1 MATLAB Code of ODE Model

5.1.1 MATLAB Code of Non-rolling ODE Model

```
% In this model, MATLAB package 'ode15s' is applied:

[t,Y]=ode15s('lan1',[0 2000],[.1 0 0 0 0 0]);
plot(t,Y);
xlabel('Seconds');
ylabel('Product P');

function dE= lan1(t,E)
k1=68000;
%k1=.01;
km1=.0001;
km0=.0001;
k0=1.4;
km2=.1;
k2=.94*km2;
k3=1;
km3=.0001;
k4=1;
km4=.0001;
kcat=.027;
% 1 E 2 Ea 3 Eas 4 P 5 Et 6 Es
if (t>=1000) % add enzyme in the middle of the experiment
```

```
  E(1)=.2;
end
dE = zeros(6,1);
dE(1)=0;% total number of enzyme
Esol=E(1)-E(2)-E(3)-E(5)-E(6);% enzyme in the solution

dE(2)=-k2*E(2)+km2*E(3)+k1*Esol;
dE(5)=-km3*E(5)+k3*Esol;
dE(6)=-km4*E(6)+k4*Esol;
dE(3)=-km2*E(3)+k2*E(2)+k0*Esol-km0*E(3);
dE(4)=kcat*E(3)*(1-E(4));% Product
```

5.1.2 MATLAB Code of Rolling ODE Model

```
% Applied MATLAB 'ode15s'package
% Combined with rolling process
[t,Y]=ode15s('rollingodefunction',[0 10000],[1 0 0 0 0 0]);
plot(t,Y);
xlabel('Seconds');
ylabel('Product P');

function dE= rollingodefunction(t,E)
k1=68000/6;  % divide by 6 since it is for all in solution to membrane
           % (number from bell)
%k1=.01;
km1=.0001;
km0=.0001;
k0=1.4;
```

```
km2=.1;  % rate Ea to Eas
k2=.94*km2; % rate Eas to Ea
%k3=1;% rate E to Et
k3=k1;% rate E to Et
km3=.0001; % rate Et to E
%k4=1;% rate E to Es should be 4 times larger than k3=k2
k4=4*k1;% rate E to Es should be 4 times larger than k3=k2
km4=.0001;% rate Es to E
kcat=.027;
r=2.;% rolling rate
dE = zeros(6,1);
if (t>=5000)
  E(1)=2.;
end

dE(1)=0;% Etotal
Esol=E(1)-E(2)-E(3)-E(5)-E(6);% enzyme in the solution
dE(2)=-k2*E(2)+km2*E(3)+k1*Esol+.25*r*E(6)-r*E(2);% enzyme on Side A
dE(5)=-km3*E(5)+k3*Esol-r*E(5)+.25*r*E(6);% enzyme on side T
dE(6)=-km4*E(6)+k4*Esol-.5*r*E(6)+r*E(5)+r*E(2);% enzyme on side S
dE(3)=-km2*E(3)+k2*E(2)+k0*Esol-km0*E(3);%actively absorbed enzyme
dE(4)=kcat*E(3)*(1-E(4));%product
```

5.2 C++ Code of LGA Monte Carlo Simulation

```
%In this section a C++ code for LGA Monte Carlo simulation is applied.
%The random number generation is the Mersenne Twister
```

```
%random generater, which is attached after the LGA code.

C++ code for LGA simulation:

using namespace std;
#include <string>
using std::string;
#include "randomc.h"
#include <iostream>
#include <cfloat>
#include <fstream>
//Define size of the grid
static const int nx = 100;
static const int ny = 100;
static const int dim = 3;


int
main()
{
%Define A:
%  A[.,.,0] = 1 indicates site is occupied by enzyme
%  A[.,.,0] = 0 indicates site is not occupied by enzyme
%  A[.,.,1] = 0,1,2 indicates active, side , top respectively
%  A[.,.,2] = 0, 1 if no lipid is available, if lipid is available
  int A[nx][ny][dim];
  TRandomMersenne rg1(1283);
```

```
  int i[ny];
  int j[nx];
  int ntime = 100000;//number of loop times
  int Gamma[ntime];//define the counting vector for product P
  Gamma[0]=0;
  int gamma=0;

%Define reaction coefficients
  double k1 = 0.1667;
  double k2 = 0.6667;
  double kk1 = 1./6.;
  double kk2 = 5./6.;
  double kcat = 0.027;
  double ht = 1; // seconds
  double scale = 1; // scale factor
  double kkk1 = 0.25;
  double kkk2 = 0.75;
 % total number of enzyme
  int amtenzy = 306;
  //int amtenzy = 3000;
  ofstream *fout = new ofstream("product3.txt");%save result for graphing
  int ilipidt = 10000;

%Use Matlab randomly generate two sequences from 1 to nx-1 (save in
%'randompoints'), then read them into i[] and j[] to reorder grid points;
%therefore, when we check whether there is an enzyme on the grid or not,
%the picking order of spots is random.
```

```
    i[0]=0;
    j[0]=0;
    i[nx-1]=nx-1;
    j[ny-1]=ny-1;
 ifstream reader;
    string file1("randompoints.txt");
    reader.open(file1.c_str());
    for (int m =1; m<nx-1 ; m++){
    reader >> i[m] >> j[m];
    }
    reader.close();

%Put enzyme on the grid, and control the number of lipid
  for (int m =0; m<nx; m++)
    for (int mm = 0; mm<ny; mm++){
      for (int mmm = 0; mmm<2; mmm++){
A[m][mm][mmm]=0;
      }
      double check4 = rg1.Random();% decide whether there is a lipid
      //       if(check4<0.03){%keep the amount of lipid low
       if(check4<.03){//keep the amount of lipid high
 A[m][mm][2]=1;
 ilipidt = ilipidt -1;}
       else {A[m][mm][2]=0;}
    }

% Decide the initial side of enzyme by Monte Carlo simulation
```

```
 for(int ii=0; ii<amtenzy; ii++){  //put the enzyme on the grid
      int check2 = rg1.IRandom(0,nx-1);
      int check3 = rg1.IRandom(0,ny-1);
  int check9 = rg1.IRandom(0,1);
       while(A[check2][check3][0]!=0){
 check2 = rg1.IRandom(0,nx-1);
 check3 = rg1.IRandom(0,ny-1);
       }
       A[check2][check3][0]=1;
       double check1 = rg1.Random(); %generate a number from 0 to 1
% If the number is  smaller than k1, set to active side
if(check1 < kk1){A[check2][check3][1] = 0;}
% If the number is between k1 and k2, set to side S
else if( check1 < kk2){A[check2][check3][1] = 1;}
else {A[check2][check3][1] = 2;}%otherwise, set to side T

}

    int ncount =0;
    for (int m =0; m<nx; m++)
    for (int mm = 0; mm<ny; mm++)
      if(A[m][mm][0]==0)
ncount = ncount +1;

% time loop
  for(int k = 1; k<=ntime; k++){
   int jj = 0;
```

```
% The rolling process
% Do the left downer corner
       int ii=0;
   if( A[i[ii]][j[jj]][0] ==1 ){
% Determine the rolling directio 0-up 1-right 2-down 3-left
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();%determine which side to roll to
      if(arrow == 0 && A[i[ii]][j[jj]+1][0]==0){
      A[i[ii]][j[jj]+1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){%if originally on side S
% If the number is smaller than k1, roll to active side
      if(check1 < kkk1){A[i[ii]][j[jj]+1][1] = 0;}
% If the number is between k1 and k2, roll to side S
  else if(check1 < kkk2){A[i[ii]][j[jj]+1][1] = 1;}
  else {A[i[ii]][j[jj]+1][1] = 2;}%otherwise, roll to side T
       }
% If enzyme lands originally on side T or A, then roll to s
    else {A[i[ii]][j[jj]+1][1] = 1;}
      }
% Arrow is a random pointer, which is used to decide the rolling direction.
 if(arrow == 1 && A[i[ii]+1][j[jj]][0]==0){
      A[i[ii]+1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]+1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]+1][j[jj]][1] = 1;}
```

```
  else {A[i[ii]+1][j[jj]][1] = 2;}
    }
    else {A[i[ii]+1][j[jj]][1] = 1;}
 }
if(arrow == 2 && A[i[ii]][ny-1][0]==0){
      A[i[ii]][ny-1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][ny-1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][ny-1][1] = 1;}
  else {A[i[ii]][ny-1][1] = 2;}
       }
      else
 A[i[ii]][j[jj]-1][1] = 1;
}
if(arrow == 3 && A[nx-1][j[jj]][0]==0){
      A[nx-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[nx-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[nx-1][j[jj]][1] = 1;}
  else {A[nx-1][j[jj]][1] = 2;}
       }
      else
 A[nx-1][j[jj]][1] = 1;%if originall on side T or A, then roll to s
}
   }
```

```
% Do the periodic boundary bottom
for(int ii=1; ii<nx-1;ii++) {
if( A[i[ii]][j[jj]][0] ==1){
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][j[jj]+1][0]==0){
      A[i[ii]][j[jj]+1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][j[jj]+1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][j[jj]+1][1] = 1;}
  else {A[i[ii]][j[jj]+1][1] = 2;}
        }
    else
      A[i[ii]][j[jj]+1][1] = 1;
      }
 if(arrow == 1 && A[i[ii]+1][j[jj]][0]==0){
      A[i[ii]+1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]+1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]+1][j[jj]][1] = 1;}
  else {A[i[ii]+1][j[jj]][1] = 2;}
        }
       else
 A[i[ii]+1][j[jj]][1] = 1;
```

```
 }
if(arrow == 2 && A[i[ii]][ny-1][0]==0){
      A[i[ii]][ny-1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][ny-1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][ny-1][1] = 1;}
  else {A[i[ii]][ny-1][1] = 2;}
       }
      else
 A[i[ii]][ny-1][1] = 1;
}
if(arrow == 3 && A[i[ii]-1][j[jj]][0]==0){
      A[i[ii]-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]-1][j[jj]][1] = 1;}
  else {A[i[ii]-1][j[jj]][1] = 2;}//otherwise, roll to side T
       }
    else
      A[i[ii]-1][j[jj]][1] = 1;
   }
 }
}
      % do the right downer corner
        ii=nx-1;
```

```
if( A[i[ii]][j[jj]][0] ==1){
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][j[jj]+1][0]==0){
      A[i[ii]][j[jj]+1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][j[jj]+1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][j[jj]+1][1] = 1;}
  else {A[i[ii]][j[jj]+1][1] = 2;}
        }
       else
 A[i[ii]][j[jj]+1][1] = 1;
      }
 if(arrow == 1 && A[0][j[jj]][0]==0){
      A[0][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[0][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[0][j[jj]][1] = 1;}
  else {A[0][j[jj]][1] = 2;}
        }
       else
 A[0][j[jj]][1] = 1;
 }
if(arrow == 2 && A[i[ii]][ny-1][0]==0){
      A[i[ii]][ny-1][0]=1;
```

```
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][ny-1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][ny-1][1] = 1;}
  else {A[i[ii]][ny-1][1] = 2;}
        }
       else
 A[i[ii]][ny-1][1] = 1;
}
if(arrow == 3 && A[i[ii]-1][j[jj]][0]==0){
      A[i[ii]-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]-1][j[jj]][1] = 1;}
  else {A[i[ii]-1][j[jj]][1] = 2;}
        }
       else
 A[i[ii]-1][j[jj]][1] = 1;
}
    }

    for( int jj = 1; jj<ny-1; jj++){
     % do left boundary
      int ii = 0 ;
  if( A[i[ii]][j[jj]][0] ==1){
  int arrow = rg1.IRandom(0,3);
```

```
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][j[jj]+1][0]==0){
      A[i[ii]][j[jj]+1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][j[jj]+1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][j[jj]+1][1] = 1;}
  else {A[i[ii]][j[jj]+1][1] = 2;}
        }
       else
 A[i[ii]][j[jj]+1][1] = 1;
      }
 if(arrow == 1 && A[i[ii]+1][j[jj]][0]==0){
      A[i[ii]+1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]+1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]+1][j[jj]][1] = 1;}
  else {A[i[ii]+1][j[jj]][1] = 2;}
        }
       else
 A[i[ii]+1][j[jj]][1] = 1;
 }
if(arrow == 2 && A[i[ii]][j[jj]-1][0]==0){
      A[i[ii]][j[jj]-1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
```

```
      if(check1 < kkk1){A[i[ii]][j[jj]-1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][j[jj]-1][1] = 1;}
  else {A[i[ii]][j[jj]-1][1] = 2;}
        }
       else
 A[i[ii]][j[jj]-1][1] = 1;
}
if(arrow == 3 && A[nx-1][j[jj]][0]==0){
      A[nx-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[nx-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[nx-1][j[jj]][1] = 1;}
  else {A[nx-1][j[jj]][1] = 2;}
        }
       else
 A[nx-1][j[jj]][1] = 1;
}
      }

      % do inner part
      for( int ii = 1; ii<nx-1; ii++){
     if( A[i[ii]][j[jj]][0] ==1 ){
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][j[jj]+1][0]==0){
      A[i[ii]][j[jj]+1][0]=1;
```

```
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]][j[jj]+1][1] = 0;}
 else if(check1 < kkk2){A[i[ii]][j[jj]+1][1] = 1;}
 else {A[i[ii]][j[jj]+1][1] = 2;}
       }
      else
A[i[ii]][j[jj]+1][1] = 1;
     }
if(arrow == 1 && A[i[ii]+1][j[jj]][0]==0){
     A[i[ii]+1][j[jj]][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]+1][j[jj]][1] = 0;}
 else if(check1 < kkk2){A[i[ii]+1][j[jj]][1] = 1;}
 else {A[i[ii]+1][j[jj]][1] = 2;}
       }
      else
A[i[ii]+1][j[jj]][1] = 1;
}
if(arrow == 2 && A[i[ii]][j[jj]-1][0]==0){
     A[i[ii]][j[jj]-1][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]][j[jj]-1][1] = 0;}
 else if(check1 < kkk2){A[i[ii]][j[jj]-1][1] = 1;}
 else {A[i[ii]][j[jj]-1][1] = 2;}
```

```
        }
       else
 A[i[ii]][j[jj]-1][1] = 1;
}
if(arrow == 3 && A[i[ii]-1][j[jj]][0]==0){
      A[i[ii]-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]-1][j[jj]][1] = 1;}
  else {A[i[ii]-1][j[jj]][1] = 2;}
       }
       else
 A[i[ii]-1][j[jj]][1] = 1;
}
}%end if1
      }
      % do the right boundary
         ii = nx-1;
 if( A[i[ii]][j[jj]][0] ==1 ){
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][j[jj]+1][0]==0){
      A[i[ii]][j[jj]+1][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][j[jj]+1][1] = 0;}
```

```
 else if(check1 < kkk2){A[i[ii]][j[jj]+1][1] = 1;}
 else {A[i[ii]][j[jj]+1][1] = 2;}
       }
   else
     A[i[ii]][j[jj]+1][1] = 1;
     }
if(arrow == 1 && A[0][j[jj]][0]==0){
     A[0][j[jj]][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[0][j[jj]][1] = 0;}
 else if(check1 < kkk2){A[0][j[jj]][1] = 1;}
 else {A[0][j[jj]][1] = 2;}
       }
      else
A[0][j[jj]][1] = 1;
}
if(arrow == 2 && A[i[ii]][j[jj]-1][0]==0){
     A[i[ii]][j[jj]-1][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]][j[jj]-1][1] = 0;}
 else if(check1 < kkk2){A[i[ii]][j[jj]-1][1] = 1;}
 else {A[i[ii]][j[jj]-1][1] = 2;}
       }
      else
A[i[ii]][j[jj]-1][1] = 1;
```

```
}
if(arrow == 3 && A[i[ii]-1][j[jj]][0]==0){
      A[i[ii]-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]-1][j[jj]][1] = 1;}
  else {A[i[ii]-1][j[jj]][1] = 2;}
       }
       else
 A[i[ii]-1][j[jj]][1] = 1;
}
  }%end if
    }%end for


  % do the left upper corner
      jj=ny-1;
       ii=0;
   if( A[i[ii]][j[jj]][0] == 1  ){
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][0][0]==0){
      A[i[ii]][0][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][0][1] = 0;}
```

```
 else if(check1 < kkk2){A[i[ii]][0][1] = 1;}
 else {A[i[ii]][0][1] = 2;}
       }
      else
A[i[ii]][j[jj]+1][1] = 1;
     }
if(arrow == 1 && A[i[ii]+1][j[jj]][0]==0){
     A[i[ii]+1][j[jj]][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]+1][j[jj]][1] = 0;}
 else if(check1 < kkk2){A[i[ii]+1][j[jj]][1] = 1;}
 else {A[i[ii]+1][j[jj]][1] = 2;}
       }
      else
A[i[ii]+1][j[jj]][1] = 1;
}
if(arrow == 2 && A[i[ii]][j[jj]-1][0]==0){
     A[i[ii]][j[jj]-1][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]][j[jj]-1][1] = 0;}
 else if(check1 < kkk2){A[i[ii]][j[jj]-1][1] = 1;}
 else {A[i[ii]][j[jj]-1][1] = 2;}
       }
      else
A[i[ii]][j[jj]-1][1] = 1;
```

```
}
if(arrow == 3 && A[nx-1][j[jj]][0]==0){
      A[nx-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[nx-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[nx-1][j[jj]][1] = 1;}
  else {A[nx-1][j[jj]][1] = 2;}
       }
      else
 A[nx-1][j[jj]][1] = 1;//if originall on side T or A, then roll to s
}
   }



    % do the top boundary
   jj = ny-1;
for(int ii=1; ii<nx-1;ii++) {
  if( A[i[ii]][j[jj]][0] ==1 ){
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][0][0]==0){
      A[i[ii]][0][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][0][1] = 0;}
```

```
 else if(check1 < kkk2){A[i[ii]][0][1] = 1;}
 else {A[i[ii]][0][1] = 2;}
       }
      else
A[i[ii]][0][1] = 1;
     }
if(arrow == 1 && A[i[ii]+1][j[jj]][0]==0){
     A[i[ii]+1][j[jj]][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]+1][j[jj]][1] = 0;}
 else if(check1 < kkk2){A[i[ii]+1][j[jj]][1] = 1;}
 else {A[i[ii]+1][j[jj]][1] = 2;}
       }
      else
A[i[ii]+1][j[jj]][1] = 1;
}
if(arrow == 2 && A[i[ii]][j[jj]-1][0]==0){
     A[i[ii]][j[jj]-1][0]=1;
 A[i[ii]][j[jj]][0]=0;
   if(A[i[ii]][j[jj]][1]==1){
     if(check1 < kkk1){A[i[ii]][j[jj]-1][1] = 0;}
 else if(check1 < kkk2){A[i[ii]][j[jj]-1][1] = 1;}
 else {A[i[ii]][j[jj]-1][1] = 2;}
       }
      else
A[i[ii]][j[jj]-1][1] = 1;
```

```
}
if(arrow == 3 && A[i[ii]-1][j[jj]][0]==0){
      A[i[ii]-1][j[jj]][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]-1][j[jj]][1] = 1;}
  else {A[i[ii]-1][j[jj]][1] = 2;}
       }
      else
 A[i[ii]-1][j[jj]][1] = 1;
}
  }//end if1
}//end for


   % do the right upper corner
 ii=nx-1;
       if( A[i[ii]][j[jj]][0] ==1){
  int arrow = rg1.IRandom(0,3);
      double check1= rg1.Random();
      if(arrow == 0 && A[i[ii]][0][0]==0){
      A[i[ii]][0][0]=1;
  A[i[ii]][j[jj]][0]=0;
    if(A[i[ii]][j[jj]][1]==1){
      if(check1 < kkk1){A[i[ii]][0][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][0][1] = 1;}
```

```
  else {A[i[ii]][0][1] = 2;}
        }
    else
      A[i[ii]][0][1] = 1;
      }
      if(arrow == 1 && A[0][j[jj]][0]==0){
A[0][j[jj]][0]=1;
A[i[ii]][j[jj]][0]=0;
if(A[i[ii]][j[jj]][1]==1){
  if(check1 < kkk1){A[0][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[0][j[jj]][1] = 1;}
  else {A[0][j[jj]][1] = 2;}
}
else
  A[0][j[jj]][1] = 1;
 }
      if(arrow == 2 && A[i[ii]][j[jj]-1][0]==0){
A[i[ii]][j[jj]-1][0]=1;
A[i[ii]][j[jj]][0]=0;
if(A[i[ii]][j[jj]][1]==1){
  if(check1 < kkk1){A[i[ii]][j[jj]-1][1] = 0;}
  else if(check1 < kkk2){A[i[ii]][j[jj]-1][1] = 1;}
  else {A[i[ii]][j[jj]-1][1] = 2;}
}
else
  A[i[ii]][j[jj]-1][1] = 1;
      }
```

```
      if(arrow == 3 && A[i[ii]-1][j[jj]][0]==0){
A[i[ii]-1][j[jj]][0]=1;
A[i[ii]][j[jj]][0]=0;
if(A[i[ii]][j[jj]][1]==1){
  if(check1 < kkk1){A[i[ii]-1][j[jj]][1] = 0;}
  else if(check1 < kkk2){A[i[ii]-1][j[jj]][1] = 1;}
  else {A[i[ii]-1][j[jj]][1] = 2;}
}
else
  A[i[ii]-1][j[jj]][1] = 1;
      }
}//end if1
//}//end if
  //}//end for s
% End of the rolling process

% Reaction Process:

 ncount =0;
    for (int m =0; m<nx; m++)
    for (int mm = 0; mm<ny; mm++)
      if(A[m][mm][0]==0)
ncount = ncount +1;

% n is applied to count the number of enzyme that we lost in reaction
  int n = 0;
  int itracker[nx*ny][2];
```

```
 int icounter = 0;
 int nncount = 0;
 for(int ii=0; ii<nx; ii++){
  for(int jj=0; jj<ny; jj++){
    if(A[i[ii]][j[jj]][0]==1){
      if(A[i[ii]][j[jj]][1]==0){//grid occupied by enzyme with active side
// itracker[icounter][0]=i[ii];
//  itracker[icounter][1]=j[jj];
//  icounter=icounter+1;
 double check =rg1.Random();
% If the grid is occupied by an enzyme with active side
and there is an available lipid on that spot
 if(A[i[ii]][j[jj]][2]==1&& check < kcat){
   A[i[ii]][j[jj]][2] = 0; // the lipid is consumed
   if(ilipidt >0){
     int itmpx  =rg1.IRandom(0,nx-1);
     int itmpy  =rg1.IRandom(0,ny-1);
     cout<<" what "<<ilipidt<<" "<<itmpx<<" "<<itmpy<<"\n";

     while( A[itmpx][itmpy][2] == 1){
       itmpx  =rg1.IRandom(0,nx-1);
       itmpy  =rg1.IRandom(0,ny-1);
     }
     A[itmpx][itmpy][2] = 1; // add lipid
     ilipidt = ilipidt -1;
   }
   n = n+1;
```

```
   A[i[ii]][j[jj]][0] = 0;//this spot becomes empty

   itracker[icounter][0]=i[ii];
   itracker[icounter][1]=j[jj];
          icounter=icounter+1;
 }
       }
     }
     else{// no enzyme at grid
       itracker[icounter][0]=i[ii];
       itracker[icounter][1]=j[jj];
       icounter=icounter+1;
     }
   }
  }
    int nn = n + nncount;

% The enzyme is added in the middle of the reaction
//     if(k==50000)
//     nn=nn+amtenzy;

 for(int ii=0; ii<nn; ii++){  //keep the number of enzyme on the grid constant
   //          int check9 = rg1.IRandom(0,1);
      int check2 = rg1.IRandom(ii,icounter-1);
      // if(check9<refill){
       A[itracker[check2][0]][itracker[check2][1]][0]=1;
       double check1 = rg1.Random(); //generate a number from 0 to 1
```

```
% If the number is  smaller than k1, set to active side
if(check1 < kk1){A[itracker[check2][0]][itracker[check2][1]][1] = 0;}
% If the number is between k1 and k2, set to side S
else if( check1 < kk2){A[itracker[check2][0]][itracker[check2][1]][1] = 1;}
else {
  A[itracker[check2][0]][itracker[check2][1]][1] = 2;
   }//otherwise, set to side T
// now make sure we do not use the same site again
int jtmp0 = itracker[ii][0];
int jtmp1 = itracker[ii][1];
itracker[ii][0] = itracker[check2][0];
itracker[ii][1] = itracker[check2][1];
itracker[check2][0] = jtmp0;
itracker[check2][1] = jtmp1;
 }
   }
 ncount =0;
    for (int m =0; m<nx; m++)
    for (int mm = 0; mm<ny; mm++)
      if(A[m][mm][0]==0)
ncount = ncount +1;

% gamma is the total number of product P.
In order to graph the amount of product with time, we saved it in a vector
Gamma[K]

gamma = n + gamma;
```

```
Gamma[k] = gamma;
 *fout<<k*ht<<" "<<Gamma[k]/scale<<"\n";
  }// end time loop
  fout->close();
  delete fout;
}
```