

國立臺灣大學生命科學院生化科學研究所



碩士論文

Graduate Institute of Biochemical Sciences

College of Life Science

National Taiwan University

Master Thesis

朗之萬方程式在蛋白質瞬間增量模型之應用

The Langevin approach for protein burst production

顏兆銘

Chao-Ming Yen

指導教授：陳水田 博士

許昭萍 博士

Advisor: Shui-Tein Chen, Ph.D.

Chao-Ping Hsu, Ph.D.

中華民國 102 年 7 月

July, 2013

誌謝



能夠完成這篇論文，首先我想感謝我的父母，他們是這篇論文最大的幕後功臣。因為有他們無私的付出和支持，使我得以專心完成學業。然後我想感謝我的指導教授陳水田老師。陳老師鼓勵我們自主思考，讓我有機會跳脫以往微觀生化的框架，以系統的角度去看待複雜的生命現象。再來我想感謝我的另一位指導教授許昭萍老師，許老師願意讓一位沒有理論背景的學生來參與電腦模擬的研究，讓我得以用數學模型一窺生物系統的奧妙。雖然相處不到一年，但我學到了很多，這都歸功於許老師願意耐心地解釋基本觀念並同時包容我的鑽牛角尖。然後我想感謝許老師理論物化實驗室裡生物組的成員們：陳奕丞、Surendhar Reddy Chepyala、江蘇峰和顏清哲，謝謝你們總是適時地為我解惑；尤其是清哲學長，總是能讓我問完一些怪問題後又帶著滿意的答案離開。最後，我想謝謝 Amine Maaref 和 Ramesh Annavajjala 在 IEEE Communications Letters 上發表的論文，它給了我莫大的啟發，解決了我在進行電腦模擬時所碰到的問題，進而催生了這篇論文，謝謝你們。



中文摘要



基因的表達中牽涉了許多隨機的化學分子碰撞與交互作用。而這些隨機的特性反映在蛋白質上，造成量的不確定性且上下波動，類似雜訊。這些雜訊在近年來的研究被證實對生物系統有絕對的重要性，舉凡幹細胞分化、或者噬菌體在裂解(lytic)與溶源(lysogenic)間的決策行為，都與是生物體利用雜訊的現象。而這些基因雜訊的研究皆可以透過數學模型，或是利用電腦模擬隨機過程來獲得可靠的結論。在本論文中，我們利用 Gillespie 演算法來模擬最近提出的蛋白質瞬間增量(burst production)的現象，然後我們利用統計理論推導出朗之萬方程式(Langevin equation)逼近 Gillespie 演算法所需要的正確雜訊強度。我們發現在蛋白質瞬間增量的情況下，朗之萬方程式的雜訊強度是沒有瞬間增量的 $\sqrt{2}$ 倍。這個結果讓我們在研究基因調控網路中的雜訊行為時，可以用朗之萬方程式來取代原本的 Gillespie 演算法，並利用朗之萬方程式可以分離雜訊的特點，去追縱調控網路中各個基因對雜訊的貢獻。

關鍵字：基因雜訊、Gillespie 演算法、朗之萬方程式、burst production、基因調控網路

ABSTRACT



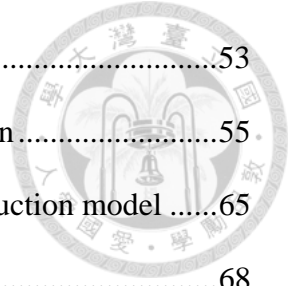
The gene expression is inherently stochastic, and can be characterized by the distribution of protein levels in individual cells. This stochasticity has been discussed analytically and modeled in stochastic process. In this thesis, we model the recently observed burst effect in protein production and use the Gillespie algorithm for simulation. Later, we derive the corresponding Langevin equation mathematically, and we found that the noise size in burst case is $\sqrt{2}$ times of non-burst case. Hence we propose a numerical Langevin approach based on stochastic process for gene regulatory network modeling. This approach is capable of describing the burst effect, and it produces the same noise as the traditional Gillespie algorithm, which is an exact realization of the master equation. Furthermore, since it is possible to partition the noise in Langevin equation, this approach has an advantage of studying the noise behavior in specific gene regulatory network.

CONTENTS



口試委員會審定書	#
誌謝	i
中文摘要	ii
ABSTRACT	iii
CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	ix
Chapter 1 Introduction.....	1
Chapter 2 The Stochastic Process and the Master Equation.....	4
Chapter 3 Numerical Approaches for Simulating Stochastic Process.....	7
3.1 The Gillespie algorithm	7
3.2 The Langevin equation	18
3.3 Approximation of Gillespie algorithm with the Langevin equation.....	22
3.3.1 Condition 1: The tau-leaping condition in Gillespie algorithm	23
3.3.2 Condition 2: Approximation with a normal random variable	25
Chapter 4 The Two-step of Single Gene Expression.....	29
4.1 Numerical simulation result: Gillespie algorithm	31
4.2 Failure in Langevin approximation to Gillespie algorithm in single gene expression model	37
Chapter 5 The Burst Production Model of Single Gene Expression.....	45
5.1 Modified Gillespie algorithm in burst production model.....	50
5.2 The Langevin form of burst production.....	53

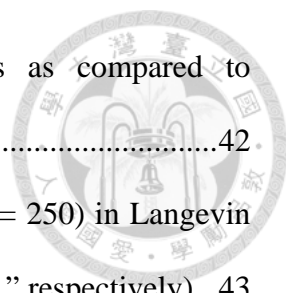
5.2.1	The Erlang distribution	53
5.2.2	Derivation of the Langevin form of burst production.....	55
5.3	Numerical Validation of the Langevin form in burst production model	65
REFERENCE		68



LIST OF FIGURES



Figure 2-1 One-dimensional random walk with discretized state.....	5
Figure 3-1 Three stochastic trajectories generated by Gillespie SSA.....	16
Figure 3-2 The steady state distribution of reactant species S in Eq.(3.19).	17
Figure 3-3 Comparison of the Gillespie algorithm and the Langevin equation.	21
Figure 4-1 Two-step model for central dogma.	29
Figure 4-2 A sample stochastic trajectory of mRNA (top) and protein (bottom). Shown are stochastic trajectories simulated by Gillespie algorithm with parameters listed in Table 4-1.	34
Figure 4-3 Steady state distribution of mRNA in single expression model.	35
Figure 4-4 Steady state distribution of protein in single expression model.	35
Figure 4-5 Approximating Gillespie algorithm with different size of dt in Langevin. ...	38
Figure 4-6 Langevin trajectory for mRNA (green) with different size of dt , compared to the stochastic trajectory of Gillespie (red) and deterministic ODE solution (dashed line). Parameters used in simulation are listed in Table 4-1 of section 4.1.	39
Figure 4-7 Langevin trajectories for protein with different size of dt , compared to the trajectory of Gillespie algorithm and deterministic ODE solution. Parameters used in simulation are listed in Table 4-1 of section 4.1.....	40
Figure 4-8 The errors of Langevin simulation as compared to analytical results, different combination of parameters (k_m, dt) is used for scanning. Shown are error in Mean (upper panels) and in Fano factor (lower panels) for traditional Langevin (Eq. (4.9)). The error are the difference between the	



statistic values of proteins from 10,000 trajectories as compared to analytical results at steady state (Eq. (4.8) in section 4.1).....42

Figure 4-9 Approximating Gillespie algorithm with ($k_m = 0.006$, $dt = 250$) in Langevin equation. (M denotes “mean” and F denotes “Fano factor,” respectively)...43

Figure 4-10 Particle number distribution in mRNA (left) and Protein (right) at a higher expression rate with parameters ($k_m = 0.08$, $dt = 50$), the figure shows that Langevin approximation can be valid in a higher expression case.44

Figure 5-1 Transformation of the two-step gene expression model into the translational burst model.47

Figure 5-2 Translational burst production model for central dogma.....50

Figure 5-3 Sample trajectories of Gillespie in two-step model and burst model. The parameters used in simulation are listed in Table 4-1 of section 4.1.52

Figure 5-4 Steady state distributions of protein in two-step model and in burst model. Shown are distributions from 10,000 trajectories simulated with the parameters listed in Table 4-1 of section 4.1. The simulation results of both models in Gillespie algorithm are quite the same.....52

Figure 5-5 Illustrated queuing problem.....54

Figure 5-6 A schematic description for two-step model (left) or a bursting one-step model (right), and for Gillespie algorithm (upper panels) and their corresponding Langevin equation (lower panels).....64

Figure 5-7 A sample trajectory for Langevin simulation in burst model. The parameters used in simulation are listed in Table 4-1 of section 4.1. The time step used in simulation is 150 sec.65

Figure 5-8 Steady state distribution of protein in different simulation approaches. Shown are distributions from 10,000 trajectories simulated with the

parameters listed in Table 4-1 of section 4.1. The time step used in both Langevin forms is 150 sec.66

Figure 5-9 Comparison of different error levels in two different Langevin forms. Shown are error in Mean (upper panels) and in Fano factor (lower panels) for traditional Langevin (Eq. (3.37) in section 3.3.2 or Eq. (4.9) in section 4.2, panels to the left) and the Burst-Langevin (Eq. (5.32) in section 5.2.2, panels to the right) in simulating a single gene expression model with the parameters listed in Table 4-1. The error are the difference between the statistic values of proteins from 10,000 trajectories as compared to analytical results at steady state (Eq. (4.8) in section 4.1).....67



LIST OF TABLES

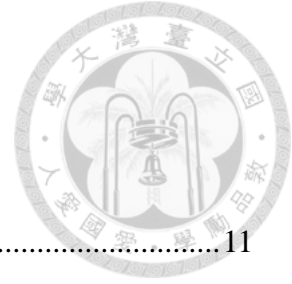


Table 3-1	An intuitive SSA.....	11
Table 3-2	Standard procedures for Gillespie SSA	14
Table 3-3	The τ -leap condition.....	24
Table 4-1	Reaction rate constants for single gene expression	32
Table 4-2	Probability rate constants for single gene expression.....	33
Table 5-1	Statistical quantities derived from Gillespie simulation and analytical Gamma distribution.	49

Chapter 1 Introduction



Gene expression is a series of biochemical reactions, each involves a different degree of randomness. The stochastic fluctuation in gene expression can result in uncertainties of protein concentration, and leads to cell-to-cell variations, or heterogeneities [1]. Moreover, this phenomenon is ubiquitous and has been demonstrated through dual-fluorescence experiments in different biological systems such as *E. coli* [2], *C. elegans* [3], and mouse [4]. The gene expression noise can arise from multiple sources [5]. For gene expression level, it could be the different particle numbers of mRNA/protein degradation machinery, or different nucleosome occupancy, or even the different nuclear architecture. For other extrinsic factors, it could be the fluctuation in microenvironment, or different partitioning after cell division, or it could be some pathway-specific propagation, where the intrinsic noise from upstream regulatory gene becomes the extrinsic noise of the downstream gene through regulation [5]-[8]. The noise in gene expression is not just an unwanted effect; instead, it has important functional role in biological system. Since many biochemical processes involves low copy number, this makes cell more susceptible to noise, and it leads to a possibility that heterogeneity in phenotype could be the consequence of different level of fluctuation in genetic network [9].

One of the key functional advantages of noise is the ability to enable probabilistic differentiation of otherwise identical cells. For example, the stochastic mating-type switch in *Saccharomyces cerevisiae*; the stochastic choice of cell type in asymmetric cell division processes of *Drosophila melanogaster* [10]; or the fluctuation in Nanog expression of mouse embryonic stem cells, which leads to stochastic cell fate decision [4], [9]. Another noise-driven cellular phenomenon is the stochastic state-switching in

the context of physiology, development, stress response, cancer and pathogenicity [11]. For example, the switch between lysogenic and lytic state in bacteriophage λ is mediated through the molecular fluctuation of the repressor CI [12].

Cells may determine their fates by playing dice with controlled odds [13]. For example, increasing evidences have shown that cells evolved strategies to control, or to exploit this inherent stochasticity, through the specific architecture of *gene regulatory networks* (GRN) [1], [5], [9]. Such interactions could include transcriptional regulation and post-translational modifications. For example, the stochastic-state switching systems are often based on positive feedback loops, which amplify the noise and leads to a bimodal distribution in extreme cases; conversely, and negative feedback loops tends to attenuate the noise and narrow the distribution to uniform the cells [14]. Hence cells in differential process tend to maintain the variability and hence prefer the noise-amplifying circuits, while cells in mature state might choose noise-filtering circuits to maintain the stability [9].

As the rapid development of biological technology has provided evidence for stochasticity in gene expression, computer simulation and mathematical analysis on the dynamics can further help to confirm these architecture-dependent noise, or predict the noise of members in the interested gene circuits. To describe the stochasticity in gene expression properly, we need to model the gene networks in the stochastic way. Reliable numerical approaches are needed for generating proper size of noise. Two typical approaches are used in most of gene expression noise related articles: the Gillespie algorithm, and the Langevin equation. Although the Gillespie algorithm provides a more accurate realization of stochastic process, it is not capable of simulating a large network because of the computational cost is high. Conversely, the Langevin equation seems to be a better choice, since much fewer computational steps are needed and one can afford

a larger-scale simulation. Moreover, the most important advantage of Langevin equation is the ability to partition the noise (as will be discussed in Section 3.2).

In this thesis, we propose a new formula for describing gene expression noise in the form of Langevin equation, which approximates the Gillespie algorithm in simulating a given single gene expression model that has the translational “bursting” properties (discussed in chapter 5), and we found the Langevin equation with this new burst production noise is more robust for simulating a single gene expression. Our result provides a better approximation to Gillespie algorithm, compared to the traditional approach discussed in Ref. [19].

Chapter 2 The Stochastic Process and the Master Equation

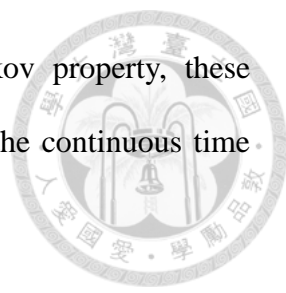


A *stochastic process*, which is a collection of random variables, can often be modeled as a *random walk*. When applied in the dynamic description, it becomes a sequence of random variables of time. The concept of random walk comes from the famous Brownian motion, which is so unpredictable and full of randomness. To understand the stochastic process, it's convenient to start from a simple example of one-dimensional random walk [15], [16]. Consider a “drunken” pedestrian that is restricted in one-dimension, and we can discretize the line so that each position equals to a unique state. The set of all possible state is then a state space. In order to make sure it is purely random, suppose the probability of moving forward and backward depends only on the “current state” of the pedestrian; in other words, for current state, the historical behavior has nothing to do with the decision of next move. This memoryless property also called the *Markov property*, which has been assumed very often in stochastic modeling, and this memoryless random walk also termed a *Markov process*.

The probability that our drunken pedestrian is in a particular state at some time, can be described as

$$P(S_N, t + \tau) = \sum_i W(S_N, t + \tau | S_i, t) P(S_i, t). \quad (2.1)$$

Here, $P(S_N, t + \tau)$ means the probability that the pedestrian is in state S_N at time $t + \tau$, and it depends on where it was previously, or $P(S_i, t)$, multiplied by the *transition probability* W of walking to state S_N from state S_i during the period



lasting from t to $t + \tau$. According to the assumption of Markov property, these transition probabilities W are constant functions of time. Taking the continuous time limit, the differential form of Eq. (2.1) becomes

$$\begin{aligned} \frac{dP}{dt} = & W'(S_N | S_{N-1})P(S_{N-1}) + W'(S_N | S_{N+1})P(S_{N+1}) \\ & - [W'(S_N | S_{N-1}) + W'(S_N | S_{N+1})]P(S_N). \end{aligned} \quad (2.2)$$

The Eq. (2.2) is the *master equation* of this one-dimensional stochastic process, which describe the dynamics of the probability. The transition probabilities W have been replaced by *transition probabilities per unit time* W' , a more general form of higher dimension will be discussed in the next chapter. The transition rate of forward walking $W'(S_N | S_{N-1})$ equals to $W'(S_{N+1} | S_N)$ in a typical Markov process, and it will be a constant value independent of the state. Same argument can be made in backward walking, hence $W'(S_N | S_{N+1})$ equals to $W'(S_{N-1} | S_N)$ in Eq. (2.2).

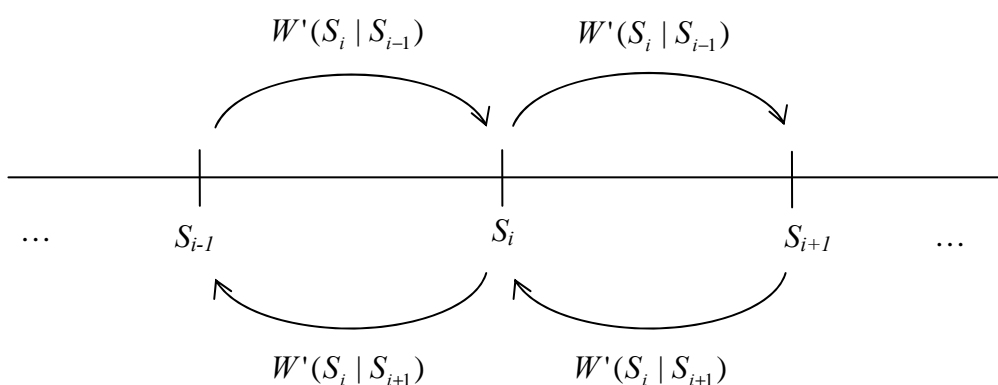
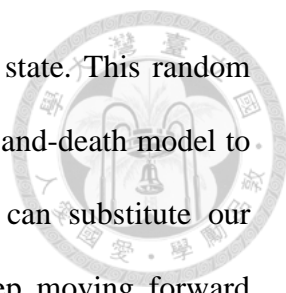


Figure 2-1 One-dimensional random walk with discretized state.



In Figure 2-1, S_i denotes different state, and i is the index of state. This random walk is fully stochastic and can be used to construct a random birth-and-death model to describe the fluctuation in a reactant species. For example, we can substitute our drunken pedestrian with an interested particle, say, mRNA. A step moving forward means one more particle is produced; conversely, a step moving backward means one particle is degraded. We can also introduce more than one species to construct a higher dimensional random walk, and it will be just like a Brownian motion in state space! In other words, in simulating for a biochemical system, the reactions take place according to their reaction probabilities (determined by the corresponding reaction rates), and the “state” of a system is the particle numbers of all relevant molecules.

In general, it is difficult to solve the master equation analytically in order to realize a stochastic process. Instead, it is usually “solved” in a numerical way by simulating the process with Monte Carlo approach, which collects a large amount of stochastic trajectories and computes the statistics. There are two frequently used numerical approach for describing a stochastic process: the *Gillespie stochastic simulation algorithm* (Gillespie SSA, or simply Gillespie algorithm) [17]-[19], and the *Langevin equation*. The Gillespie algorithm is a discrete, and particle number-based stochastic simulation, which can be used to simulate the stochasticity of chemical kinetics in molecular level. Besides, the simulation result of Gillespie algorithm is an exact realization of the master equation, which means there is no any approximation in the result. The Langevin equation is another way to describe stochastic process. It is a differential equation with additional random variables. More details on the Gillespie algorithm and Langevin equation are given in the next chapter.

Chapter 3 Numerical Approaches for Simulating Stochastic Process



3.1 The Gillespie algorithm

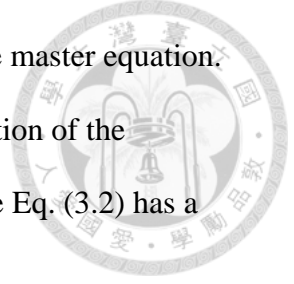
Consider a system of fixed volume Ω contains N different species of reactants, with M different reactions among those reactants, or *reaction channels*, denotes R_j ($j = 1, \dots, M$). And assume particle number of each reactant can be expressed as a continuous function of time $X_i(t)$ ($i = 1, \dots, N$). For deterministic process, the dynamics of the system can be described by the following ordinary differential equation (O.D.E.) form:

$$\begin{aligned}
 \dot{X}_1 &= f_1(X_1, \dots, X_N) \\
 \dot{X}_2 &= f_2(X_1, \dots, X_N) \\
 &\dots \\
 \dot{X}_N &= f_N(X_1, \dots, X_N)
 \end{aligned}
 \tag{3.1}$$

For simplification, we use the *state vector* $\mathbf{X}(t) \equiv (X_1(t), X_2(t), \dots, X_N(t))$ to describe the system state at time t . In stochastic process, $X_i(t)$ is actually a random variable. To analyze it, define *propensity function* a_j for reaction channel R_j , which is that

$$\begin{aligned}
 a_j(\mathbf{x})dt &\equiv \text{the probability, given } \mathbf{X}(t) = \mathbf{x}, \\
 &\text{that one } R_j \text{ channel reaction will occur} \\
 &\text{somewhere inside } \Omega \text{ in the next infinitesimal} \\
 &\text{time interval } [t, t + dt) \quad (j = 1, \dots, M).
 \end{aligned}
 \tag{3.2}$$

Eq. (3.2) is the fundamental premise for numerically solving the master equation. Previously it was thought to be an *ad hoc* assumption for stochastization of the deterministic chemical kinetics [20]. Later it has been proved that the Eq. (3.2) has a rigorous microphysical basis [21].



For each reaction channel R_j , we also need to define *the state-change vector* \mathbf{v}_j , whose *ith* component is given by

$$v_{ji} \equiv \text{the change in the number of } X_i \text{ produced by one } R_j \text{ channel.} \quad (3.3)$$

Together, Eq. (3.2) and Eq. (3.3) specify the behavior of each reaction channel, the state-change vector \mathbf{v}_j is a fixed vector with integer value, and its component is usually +1 for production channel, and -1 for degradation channel, while the propensity function a_j changes value from time to time accords to the system state \mathbf{x} . From previous argument [17], the function a_j has mathematical form

$$a_j(\mathbf{x}) = c_j h_j(\mathbf{x}). \quad (3.4)$$

Here, c_j is the *specific probability rate constant* for channel R_j , and $c_j dt$ gives the probability that “one” random chosen pair of reactant molecules in R_j will react accordingly in the next infinitesimal time. The unit of c_j is the inverse of time, and the value can be determined from traditional reaction constant k_j . If the system’s volume Ω is given, c_j can be derived from k_j by transforming the unit of k_j into the same unit of c_j . For example, consider two reactant molecular species S_1 and S_2 , and a reaction

channel R_μ of which S_1 reacts with S_2 to give the product S_3



The reaction channel R_μ was conventionally described by reaction rate constant k_μ in chemical kinetics, more precisely, it's the average reaction rate per unit volume divided by the product of the average densities of the reactants [17]. Put the definition of c_j together, we have

$$k_\mu = \langle x_1 x_2 \rangle \Omega c_\mu / \langle x_1 \rangle \langle x_2 \rangle. \quad (3.6)$$

Where x_1 and x_2 represents the molecular concentration of reactant species S_1 and S_2 . However, in deterministic formulation we do not consider statistical variance, therefore $\langle x_1 x_2 \rangle = \langle x_1 \rangle \langle x_2 \rangle$, hence it gives the relation

$$k_\mu = \Omega c_\mu. \quad (3.7)$$

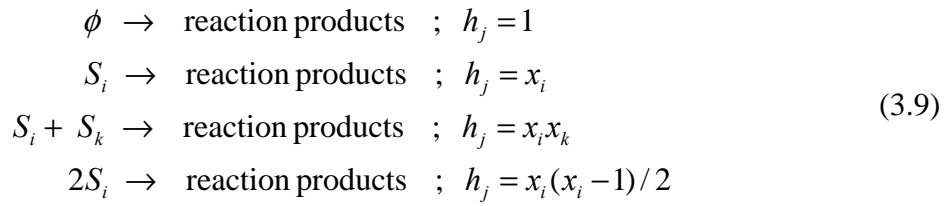
For other types of reaction channel, this relationship can vary, and we summarize it as following

$$\begin{aligned} \phi &\rightarrow \text{reaction products} ; k_\mu = c_\mu / \Omega \\ S_i &\rightarrow \text{reaction products} ; k_\mu = c_\mu \\ S_i + S_k &\rightarrow \text{reaction products} ; k_\mu = \Omega c_\mu \quad (i \neq k) \\ 2S_i &\rightarrow \text{reaction products} ; k_\mu \cong \Omega c_\mu / 2 \end{aligned} \quad (3.8)$$

For convenience, we often use unit volume in simulation. By doing so, it won't

change the value of k_μ of the first three types of reaction after we transfer it to c_μ in stochastic simulation.

The function $h_j(\mathbf{x})$ in Eq. (3.4) is defined to be the number of distinct combinations of R_j reactant molecules available in the state \mathbf{x} [19]. For example, the h_j for the following types of reaction can be



The time-dependent stochastic trajectory can thus be simulated once c_j and \mathbf{v}_j are given. Here, the evolution of Eq. (3.2) implies that the state vector $\mathbf{X}(t)$ is a jump-type Markov process non-negative N-dimensional integer lattice [19], and a traditional way to analyze it is to write it in the form of conditional probability function

$$P(\mathbf{x}, t | \mathbf{x}_0, t_0) \equiv \text{Prob}\{\mathbf{X}(t) = \mathbf{x}, \text{ given that } \mathbf{X}(t_0) = \mathbf{x}_0\}.
 \tag{3.10}$$

The time-dependent evolution of this conditional probability function can be derived by advance the system with a small time step dt one at a time, so that we can exclude the probability for two or more reactions to occur in dt . Using Eq. (3.2), together with the previous argument [21], the probability of the system being in state \mathbf{x} at time $t + dt$ can thus be described

$$\begin{aligned}
P(\mathbf{x}, t + dt | \mathbf{x}_0, t_0) \equiv & P(\mathbf{x}, t | \mathbf{x}_0, t_0) \times \left[1 - \sum_{j=1}^M a_j(\mathbf{x}) dt \right] \\
& + \sum_{j=1}^M \left[P(\mathbf{x} - \mathbf{v}_j, t | \mathbf{x}_0, t_0) a_j(\mathbf{x} - \mathbf{v}_j) dt \right]
\end{aligned} \tag{3.11}$$



If we take the limit, the differential form of Eq. (3.11) becomes the chemical master equation

$$\frac{\partial}{\partial t} P(\mathbf{x}, t | \mathbf{x}_0, t_0) \equiv \sum_{j=1}^M \left[a_j(\mathbf{x} - \mathbf{v}_j) P(\mathbf{x} - \mathbf{v}_j, t | \mathbf{x}_0, t_0) - a_j(\mathbf{x}) P(\mathbf{x}, t | \mathbf{x}_0, t_0) \right] \tag{3.12}$$

Eq. (3.12) is an exact consequence of Eq. (3.2), if we can solve for P , then we can fully understand the behavior of this stochastic process. As we have mentioned above, it is rare that an exact analytical solution to Eq. (3.12) can be found; but at least, we could generate many trajectories enough to describe the probability density of P .

To numerically simulate Eq. (3.12), an intuitive way is to give a very small time interval step dt , and iterate the following steps

Table 3-1 An intuitive SSA

Step 0	Start with system state \mathbf{x}_0 , calculate the probability $a_j(\mathbf{x}_0)dt$ for each reaction channel
Step 1	Generate <i>uniform random variables</i> , update the system state according to the probability $a_j(\mathbf{x}_0)dt$ and state change vector \mathbf{v}_j of each reaction channel R_j
Step 2	Advance the system time with dt , back to Step 0

The stochastic simulation result is an exact realization for Eq. (3.12), but the defect is that, since dt is required to be small enough, the probability $a_j(\mathbf{x}_0)dt$ that a reaction happens in reaction channel R_j will also be small, it means the computer repeat those three steps frequently, but most of time there's no reaction happened, this leads to inefficiency of this naive stochastic simulation algorithm, and makes it even unusable if $dt \rightarrow 0$.

Another way to solve this problem of inefficiency is the *Gillespie stochastic simulation algorithm*, which defines another exact consequence of Eq. (3.2), the *next-reaction density function* $p(\tau, j | \mathbf{x}, t)$ [17], [21], where

$$p(\tau, j | \mathbf{x}, t)d\tau \equiv \begin{array}{l} \text{probability that, given } \mathbf{X}(t) = \mathbf{x}, \\ \text{the next reaction in } \Omega \text{ will occur} \\ \text{in the infinitesimal time interval} \\ [t + \tau, t + \tau + d\tau), \text{ and will be an} \\ R_j \text{ reaction.} \end{array} \quad (3.13)$$

The τ in Eq. (3.13) stands for the time that elapses without any reaction occurs in the system, and $p(\tau, j | \mathbf{x}, t)d\tau$ can also be expressed as the product

$$p(\tau, j | \mathbf{x}, t)d\tau = p_0(\tau) \cdot a_j d\tau. \quad (3.14)$$

Here, $p_0(\tau)$ stands for the probability density of this τ , and it can be calculated by elementary probability argument. Since $\sum_{j=1}^M a_j(\mathbf{x})dt$ gives the probability that *some* reaction will take place in the next dt according to the definition of Eq. (3.2), we can

divide τ , say, into K equal length subintervals. Let ε denotes the subinterval where $\varepsilon = \tau/K$, K is a positive integer, the probability none of the reactions

R_1, \dots, R_M occurs in the first ε subinterval $(t, t + \varepsilon)$ is

$$\prod_{j=1}^M [1 - a_j(\mathbf{x})\varepsilon + o(\varepsilon)] = 1 - \sum_{j=1}^M a_j(\mathbf{x})\varepsilon + o(\varepsilon). \quad (3.15)$$

Where $o(\varepsilon)$ describes the probability that more than one reaction occurs in $(t, t + \varepsilon)$, and we don't really need to worry about it since this term can be very small for small ε . The right side of Eq. (3.15) is also the probability that no reaction occurs in the subsequent intervals $(t, t + 2\varepsilon)$, $(t, t + 3\varepsilon)$ and so on, since the system state \mathbf{x} remains if no reaction occurs, hence the term $\sum_{j=1}^M a_j(\mathbf{x})\varepsilon$ remains the same. For the rest of subintervals, using the same argument, we have

$$p_0(\tau) = \left[1 - \sum_{j=1}^M a_j(\mathbf{x})\varepsilon + o(\varepsilon) \right]^K. \quad (3.16)$$

For $K \rightarrow \infty$, together with $\varepsilon = \tau/K$, Eq. (3.16) is equal to

$$p_0(\tau) = \exp \left[- \sum_{j=1}^M a_j(\mathbf{x})\tau \right]. \quad (3.17)$$

Hence, after plugging Eq. (3.17) into Eq. (3.14), it becomes

$$p(\tau, j | \mathbf{x}, t) = a_j(\mathbf{x}) \exp \left[- \sum_{k=1}^M a_k(\mathbf{x}) \tau \right] \quad (3.18)$$

$$(0 \leq \tau < \infty; j = 1, \dots, M).$$



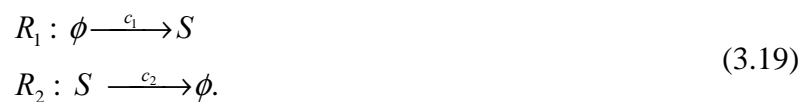
The Eq. (3.18) provides the fundamental basis for Gillespie stochastic simulation algorithm [17], in which a random pair of (τ, j) is generated by Monte Carlo procedure according the joint probability density function. The following procedure briefly describes the Gillespie algorithm [17], [18].

Table 3-2 Standard procedures for Gillespie SSA

<p>Step G₀ (initialization)</p>	<ol style="list-style-type: none"> 1. Set the reaction channels: <ul style="list-style-type: none"> • Input c_j values according to Eq. (3.8) ($j = 1, \dots, M$). • Input \mathbf{v}_j the state-change vector according to Eq. (3.3). 2. Set the reactant species: <ul style="list-style-type: none"> • Input initial values for $X_i (i = 1, \dots, N)$. 3. Set initial time $t = 0$, and simulation time t_{stop}.
<p>Step G₁</p>	<ol style="list-style-type: none"> 1. Calculate $a_j(\mathbf{x}) = c_j h_j(\mathbf{x})$ according to Eq. (3.4), where \mathbf{x} denotes the system state vector. 2. Calculate $a_0 = \sum_{j=1}^M a_j(\mathbf{x}) \tau$.
<p>Step G₂</p>	<ol style="list-style-type: none"> 1. Generate two independent <i>uniform distributed random number</i> r_1 and r_2, where $r_1 \in [0,1]$ $r_2 \in [0,1]$. 2. Determine τ: According Eq. (3.17), τ can be calculated as

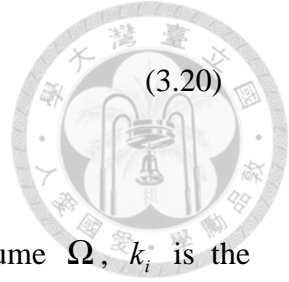
	$\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right).$ <p>3. Determine the reaction channel R_μ in which the reaction happens:</p> <p>Take μ so that $\sum_{j=1}^{\mu-1} a_j < r_2 a_0 \leq \sum_{j=1}^{\mu} a_j$.</p>
Step G ₃	<p>1. Update the simulation time</p> $t = t + \tau.$ <p>2. Update the system state according to R_μ</p> $\mathbf{x} = \mathbf{x} + \mathbf{v}_\mu.$ <p>3. If $t > t_{stop}$, or if no more reactants left (all $h_\mu = 0$), terminate the simulation; otherwise, return to Step G₁.</p>

Here, we use a simple example to demonstrate the Gillespie algorithm. Consider a simple birth-and-death model for a species S , where



Reaction channel R_1 describes a zero order constant production of species S , and R_2 describes a first order degradation of S ; c_1, c_2 is dimensionless probability rate constant, as defined in Eq. (3.8).

For conventional understanding in kinetic description, the ordinary differential equation form of this model will be



$$\frac{dx}{dt} = k_1 - k_2 x . \quad (3.20)$$

Where x denotes the concentration of S in system's volume Ω , k_i is the conventional reaction rate constant; the unit of k_1 is concentration over time, and the unit of k_2 is the inverse of time. The relationship between k_i in kinetic description and c_i in Gillespie algorithm has been discussed in Eq. (3.8). The following figure demonstrates the simulation result of the model, with the parameter $k_1 = 0.02 M / \text{sec}$ and $k_2 = 0.001 / \text{sec}$, and system's volume $\Omega = 1L$ (Hence, the corresponding $c_1 = 0.02 / \text{sec}$ and $c_2 = 0.001 / \text{sec}$)

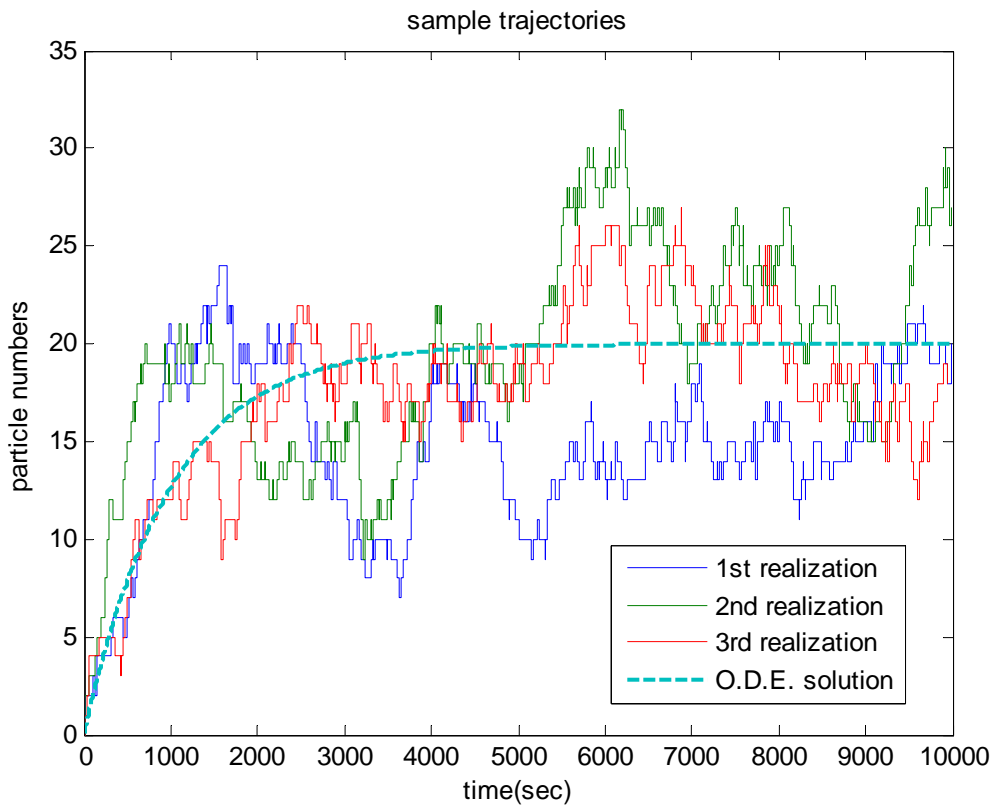


Figure 3-1 Three stochastic trajectories generated by Gillespie SSA.

Three trajectories are shown in the Figure 3-1, which are generated by repeating Gillespie algorithm three times, using the birth-and-death model in Eq. (3.19), with parameters described in text. The dashed line represent the analytical solution of Eq. (3.20), which is a continuous function with mathematical form

$$X(t) = \Omega x(t) = \Omega \frac{k_1}{k_2} (1 - e^{-k_2 t}).$$

The capital X denotes the particle number of species S , and the vertical axis shows the discrete particle number of species S . The Gillespie algorithm is programmed by MATLAB[®].

In order to reconstruct the probability space of $X(t)$, a large quantities of trajectories should be made. Here, since we are interested in the steady state X^* , we take value of each trajectory at the end of simulation time (10,000 sec in this case) and analyze statistics of the distribution.

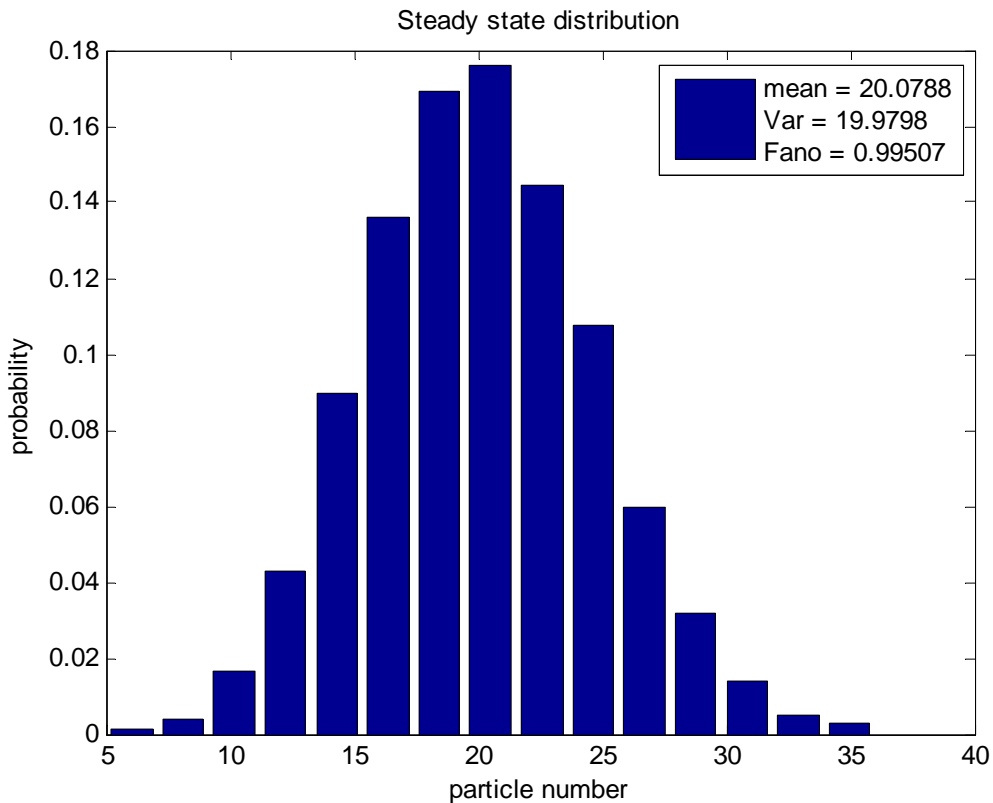


Figure 3-2 The steady state distribution of reactant species S in Eq.(3.19).

The histogram shown in Figure 3-2 is generated by steady state values (value at 10,000 sec. in this case) of 10,000 trajectories simulated with Gillespie algorithm, using the birth-and-death model described in Eq. (3.19). The statistics is made by collecting the steady state value for each trajectory. The distribution of X^* has mean = 20.08, variance = 19.98 and the Fano factor (defined as variance/mean) $\cong 1$, which represents a Poisson distribution.

3.2 The Langevin equation

The Langevin equation is a stochastic differential equation (S.D.E.), which is an ordinary differential equation with noise terms. Originally, the Langevin equation was used to describe Brownian motion, the random movement of a particle resulting from random collision in the solution. The general mathematical form of Langevin equation can be described by

$$\dot{x} = f(x) + \eta(x, t). \quad (3.21)$$

Where x is a random variable which varies with time, and $f(x)$ is conventional ordinary differential equation, describing the deterministic time-dependent behavior of x . In mechanical approach of, $f(x)$ is the sum of "regular forces" acting on the particle while x is velocity of the particle.

The noise term η is added to make the deterministic equation become stochastic, and η satisfies the following conditions:

$$\langle \eta(x, t) \rangle = 0. \quad (3.22)$$

$$\langle \eta(x, t) \eta(x, t + \tau) \rangle = q(x) \delta(\tau). \quad (3.23)$$



Here, $\delta(\tau)$ denotes the Dirac delta function and $q(x)$ is an unknown noise-generating function dependent on x , the autocorrelation function of noise equals $q(x)\delta(\tau)$ according to Eq. (3.23). The first condition is that ensemble average of the noise term equals to zero, hence if we take ensemble average of Eq. (3.21), it equals to the ensemble average of the one with only deterministic part. The second condition describes Markov property, in other words, the noise density of different step is independent of time.

For numerically simulation of Langevin equation, according to [22], the *Euler–Maruyama method* can be applied. Consider a *standard Brownian motion* (or *standard Wiener process*), where a random variable $W(t)$ defined over time interval $[0, T]$ and depends continuously on time, satisfies the following conditions:

1. $W(0) = 0$ (with probability equals 1)
2. For $[0 \leq s < t < T]$, $W(t) - W(s) \sim \sqrt{t-s}N(0,1)$ which means the increment of this stochastic process comes from a normal distribution with mean zero and variance $t-s$, $N(0,1)$ denotes the unit normal distribution.
3. For $[0 \leq s < t < u < v < T]$, the increments $W(t) - W(s)$ and $W(u) - W(v)$ are independent.

The stochastic integral of Eq. (3.21) will be the solution to the stochastic trajectory of x and it can be written as

$$x(t) = x_0 + \int_0^t f(x(s))ds + \int_0^t \eta(x(s))dW(s) \quad t \in [0, T]. \quad (3.24)$$



Here, $x_0 = x(0)$ is the initial condition, the first integral on the right side describes the historical trajectory of the deterministic part, and the second one describes the historical trajectory of the stochastic part. The differential equation form of Eq. (3.24) equals

$$dx(t) = f(x(t))dt + \eta(x(t))dW(t) \quad x(0) = x_0, \quad t \in [0, T]. \quad (3.25)$$

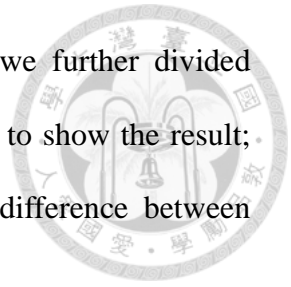
To numerically simulate Eq. (3.24), we discretize the interval into L parts. Let $\Delta t = T/L$ for some positive integer L , and let $\tau_i = i\Delta t$. The Euler–Maruyama method takes the form

$$x(\tau_i) = x(\tau_{i-1}) + f(x(\tau_{i-1}))\Delta t + \eta(x(\tau_{i-1}))(W(\tau_i) - W(\tau_{i-1})) \quad (i=1, \dots, L). \quad (3.26)$$

If we take $\eta \equiv 0$, Eq. (3.26) simply reduced to $x(\tau_i) = x(\tau_{i-1}) + f(x(\tau_{i-1}))\Delta t$, and it's the form of *Euler's method*. In other words, the fundamental concept of Euler–Maruyama method is from Euler's method, where we use the present step to calculate the increment, and then proceed to the next step according to the increment.

The following figure (Figure 3-3) shows numerical result of the Langevin equation (red), using the same model described in Eq. (3.19) and the same parameter set, compared to the result of the Gillespie algorithm (blue). We note that the Langevin equation takes continuous value, while the Gillespie algorithm displays the discrete

number of molecular particle. This problem can be bypassed if we further divided particle number with system's volume, and use the "concentration" to show the result; or if the system contains large quantity of particle number, the difference between discreteness and continuity could be sew up.



If we take away the noise term in Langevin equation, the Langevin trajectory (red line) in Figure 3-3 will exactly be the same as the trajectory of deterministic O.D.E. solution (dashed line).

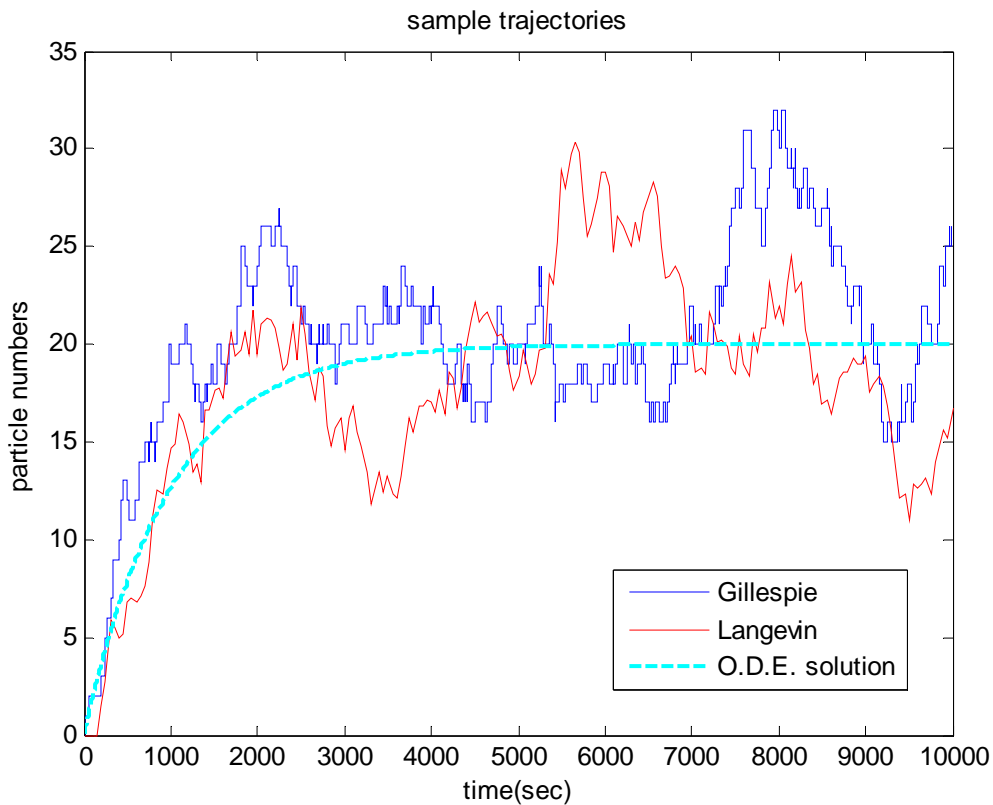


Figure 3-3 Comparison of the Gillespie algorithm and the Langevin equation.

A comparison for the trajectories derived from the Gillespie algorithm (blue line), the Langevin equation (red line), and the corresponding trajectory from the deterministic kinetics (dashed line in cyan).

3.3 Approximation of Gillespie algorithm with the Langevin equation



At first glance, the Langevin equation seems to be artificial, since one can simply manipulate the noise term to generate different noise strength. Despite the fact that one can arbitrarily tune the noise, the best thing of Langevin is the ability to separate the deterministic and the stochastic part, and it will become very useful for tracking noise source. Since the Gillespie algorithm is an exact realization of chemical master equation [17], [18], we might ask what is the proper noise size for the corresponding Langevin equation, under the same stochastic model? Before answering this question, we must first realize that these two stochastic approaches are different in nature. First, for simulation aspect, the way of describing the history axis is different. In Gillespie, the history axis of a species in the system is described in different size of waiting time (Eq. (3.13)). In Langevin, the history axis of *every* species is segmented into a uniform infinitesimal time interval dt (Eq. (3.25) and Eq. (3.26)), and the evolution of system is described by repeating this dt snapshot. Second, the Gillespie algorithm describe the discrete nature in molecular level, the reactants change numbers according to the reaction channels and the corresponding state change vectors (Eq. (3.3)), while the Langevin equation is a continuous function. Hence, if we want to make the Langevin equation “looks like” the Gillespie algorithm, the difference mentioned above must be overcome. According to the argument of Daniel T. Gillespie in [19], the Gillespie algorithm can be approximated by the Langevin equation, if dt in the Langevin equation satisfies the following two conditions.

3.3.1 Condition 1: The tau-leaping condition in Gillespie algorithm

For a typical realization of Gillespie algorithm, if we could divide the system's history axis into a set of contiguous subintervals in such a way that we could forego determine how many times each reaction channel fires in each subinterval, then it's conceivable to "leap" the history axis with these subintervals. This strategy is originally an acceleration approach for Gillespie algorithm, which is called *the τ -leap method* [23].

By notations of previous section, suppose the system's state $\mathbf{X}(t)$ at the current time t is denoted as \mathbf{x}_t . Let $K_j(\mathbf{x}_t, \tau)$ ($\tau > 0$) defined as following

$$K_j(\mathbf{x}_t, \tau) \equiv \begin{array}{l} \text{the number of times, given } \mathbf{X}(t) = \mathbf{x}, \\ \text{the reaction channel } R_j \text{ will fire in the} \\ \text{time interval } [t, t + \tau) \quad (j = 1, \dots, M). \end{array} \quad (3.27)$$

The number of reactant species S_i at time $t + \tau$ can then be expressed as

$$X_i(t + \tau) = X_i(t) + \sum_{j=1}^M K_j(\mathbf{x}_t, \tau) v_{ji} \quad (i = 1, \dots, N). \quad (3.28)$$

The second term on the right side of Eq. (3.28) means the summation of changes from each reaction channel, and v_{ji} denotes the i th component in state change vector \mathbf{v}_j (Eq. (3.3)). Under this construction, $K_j(\mathbf{x}_t, \tau)$ is a random variable, and it's difficult to compute $K_j(\mathbf{x}_t, \tau)$ just like it's hard to solve the master equation, since every reaction that takes place within τ can change the system state, and hence the propensity functions $a_j(\mathbf{x})$, since it depends on the current system state (Eq. (3.2) and Eq. (3.4)). In other words, every reaction that happens in τ is not independent of each other, the first reaction in τ will leave some "memory" on the history axis and affect the next one, unless we consider the following condition:

Table 3-3 The τ -leap condition



The τ -leap condition:

Require τ “small enough” that the change in the state during $[t, t+\tau)$ will be so slight and none of the propensity functions change its value “appreciably.”

Assume the above condition is satisfied, then during the entire interval $[t, t+\tau)$, the propensity function for each reaction channel R_j will remain “essentially constant” at the value $a_j(\mathbf{x})$ [23]. Apparently, if we make $\tau \rightarrow 0$, this condition can be easily satisfied, since most reaction channels won’t fire in such small time interval, this concept is very similar to the “intuitive SSA” which is mentioned in Table 3-1; but it will make no use for such τ leap, and we rather choose the original Gillespie algorithm. Hence, this τ should be small, but not too small.

Fortunately, if our system contains large number of population for each reactant species, the changes made by each reaction channel will be relatively small since most of state change vectors \mathbf{v}_j changes the particle number of reactants for ± 1 . We then further make τ small enough and ensure that the system state change little (and hence the propensity function) in every moment of this τ , where

$$a_j(\mathbf{x}_{t'}) \cong a_j(\mathbf{x}_t) \quad \forall t' \in [t, t+\tau), \forall j \in [1, M]. \quad (3.29)$$

In this case, all reactions that happened in τ will be essentially independent, and the random variable $K_j(\mathbf{x}_t, \tau)$ becomes the *Poisson* random variable [19], [23]:

$$K_j(\mathbf{x}_t, \tau) = \text{Poisson}(a_j(\mathbf{x}_t), \tau) \quad (j = 1, \dots, M). \quad (3.30)$$



The term *Poisson* denotes a Poisson random variable, and the mean of $\text{Poisson}(a_j(\mathbf{x}_t), \tau)$ equals to $a_j(\mathbf{x}_t)\tau$. The net effect of the τ -leaping condition allows Eq. (3.28) to be approximated by

$$X_i(t + \tau) = X_i(t) + \sum_{j=1}^M \text{Poisson}(a_j(\mathbf{x}_t), \tau) v_{ji} \quad (i = 1, \dots, N). \quad (3.31)$$

Eq. (3.31) is the τ -leaping method, and it provides a fundamental argument of the existence of such τ . Remember it's just an approximation; it means that the error could become large if the τ is too big, and the result will be close to the exact realization of Gillespie algorithm, if the τ is small enough.

3.3.2 Condition 2: Approximation with a normal random variable

Suppose we could find a set of τ which satisfies the tau-leaping condition, the Eq. (3.31) can be used to substitute the original Gillespie algorithm. If, for this set of τ , there exist a subset which further satisfies the condition where

$$\langle \text{Poisson}(a_j(\mathbf{x}_t), \tau) \rangle = a_j(\mathbf{x}_t)\tau \gg 1, \quad \forall j \in [1, M]. \quad (3.32)$$

In other words, for *every* reaction channel, the average firing times within τ should be as much as possible. Since the Poisson distribution is a description of rare

events, and the random variables of Poisson distribution are integers by definition; the condition in Eq. (3.32) provides a bridge for connecting the discrete Gillespie algorithm to continuous Langevin equation, because when mean is large, a discrete Poisson random variable can be approximated by a normal random variable, with same mean and variance [19], [23]. Where

$$Poisson(a_j(\mathbf{x}_t), \tau) \approx Normal(a_j(\mathbf{x}_t)\tau, a_j(\mathbf{x}_t)\tau) \quad \text{for } a_j(\mathbf{x}_t)\tau \gg 1, \quad (3.33)$$

Here, we use the term *Normal* to denote a normal distributed random variable. Under the condition of Eq. (3.32), the τ -leaping form of Eq. (3.31) becomes

$$X_i(t + \tau) = X_i(t) + \sum_{j=1}^M Normal(a_j(\mathbf{x}_t)\tau, a_j(\mathbf{x}_t)\tau) v_{ji} \quad (i = 1, \dots, N). \quad (3.34)$$

Notice that the increment of $X_i(t)$ is now taking continuous random variable from normal distribution. By linear combination theorem for normal random variables

$$Normal(m, \sigma^2) = m + \sigma Normal(0,1). \quad (3.35)$$

Eq. (3.34) now becomes:

$$X_i(t + \tau) = X_i(t) + \sum_{j=1}^M v_{ji} a_j(\mathbf{x}_t)\tau + \sum_{j=1}^M v_{ji} \sqrt{a_j(\mathbf{x}_t)\tau} N_j(0,1) \quad (i = 1, \dots, N). \quad (3.36)$$

Where $N_j(0,1)$ denotes the independent unit normal random variable for each reaction channel R_j . Consider a time step τ which can satisfy both conditions, and substitute this τ with the notation dt , we then rewrite the Eq. (3.36) into

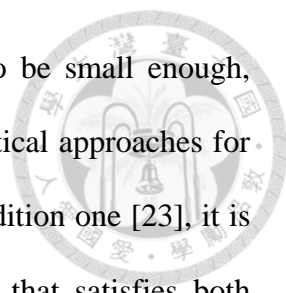
$$X_i(t+dt) = X_i(t) + \sum_{j=1}^M v_{ji} a_j(\mathbf{x}_t) dt + \sum_{j=1}^M v_{ji} \sqrt{a_j(\mathbf{x}_t)} dt N_j(0,1) \quad (i=1, \dots, N). \quad (3.37)$$

Eq. (3.37) is the working form for numerical simulation of Langevin equation, which approximates the corresponding Gillespie algorithm. The differential form of Eq. (3.37) equals to

$$\frac{dX_i(t)}{dt} = \sum_{j=1}^M v_{ji} a_j(\mathbf{X}(t)) + \sum_{j=1}^M v_{ji} \sqrt{a_j(\mathbf{X}(t))} \Gamma_j(t), \quad (3.38)$$

where $\Gamma_j(t)$ are temporally uncorrelated, statistical independent Gaussian white noise. Eq. (3.38) is exactly a Langevin equation, and the noise term is consistent with the properties described in Eq. (3.22) and Eq. (3.23). Up to this step, we have already transform the original Gillespie algorithm (which stands for the exact simulation of master equation) into the corresponding Langevin equation. The second term of Eq. (3.38) can be regarded as the noise extracted from the master equation, if we eliminate this term, Eq. (3.38) simply go back to deterministic ordinary differential equation.

The second condition requires τ in Langevin equation being “large enough” to ensure that $a_j(\mathbf{x})\tau \gg 1$, so normal distribution can approximate well to Poisson



distribution. It is interesting that the first condition requires τ to be small enough, while the second one does conversely. Even though there are analytical approaches for determining an acceptable upper bound of this τ to fulfill the condition one [23], it is still hard to propose one for analytically determining proper dt that satisfies both conditions [24]. It is clear that if τ is too large, the τ -leaping in Eq. (3.31) cannot well approximate to Gillespie algorithm; and if τ is too small, the normal random variable cannot well approximate to Poisson. The question is, we do not know which approximation is more important than the other one, or which condition we must take care first. Nevertheless, if the system contains large number of population for each reactant species, it is easier to find a proper τ which satisfies both conditions well [19]. Suppose each reactant species has large quantity in number, the system state then change little after τ -leaping. Furthermore, $a_j(\mathbf{x})\tau$ can be large under this situation, since $a_j(\mathbf{x}) = c_j h_j(\mathbf{x})$ according to Eq. (3.4). It means that the larger population, the larger $a_j(\mathbf{x})$, and hence the restriction of τ can be reduce to make $a_j(\mathbf{x})\tau \gg 1$.

According to this argument, we may at least conclude that it will be easier to use Langevin to approximate Gillespie algorithm if the system contains large number of population for each reactant species. In other words, this approximation method could be invalid if the system contains low copy number of reactants [24].

Chapter 4 The Two-step of Single Gene Expression

In cells or organisms, there is often several layers of gene regulation, forming a cascade or even a gene regulatory network. While our ultimate goal is to describe both inherent and propagated noise in a genetic network of interest, the description of noise from a single gene expression is of great fundamental importance since it offers a ground for both theoretical and computational studies. Stochastic fluctuation of biochemical processes in the central dogma can lead to expression noise of a single gene. According to [25], [26], the single gene expression can be modeled by the following linear kinetic scheme

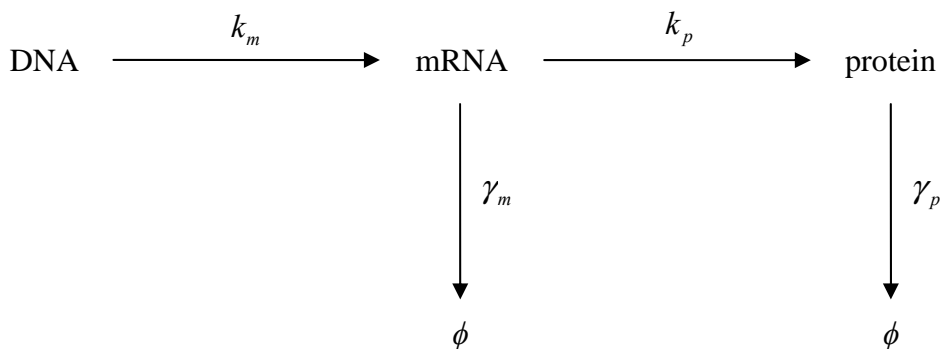
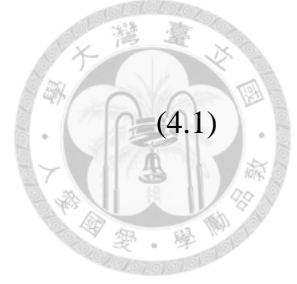


Figure 4-1 Two-step model for central dogma.

Here, k_m represents the rate of transcription of a gene into mRNA, k_p is the rate of translation of mRNA into protein, and $\gamma_{m,p}$ are the rates of degradation of mRNA and protein respectively. For deterministic ordinary differential equation, this model can be described as the following kinetic equations

$$\begin{aligned}\frac{d[mRNA]}{dt} &= k_m - \gamma_m [mRNA], \text{ and} \\ \frac{d[protein]}{dt} &= k_p [mRNA] - \gamma_p [protein].\end{aligned}\tag{4.1}$$



The steady state values of [mRNA] and [protein] can be derived from Eq. (4.1) by making $\frac{d[mRNA]}{dt} = \frac{d[protein]}{dt} = 0$, where

$$\begin{aligned}[mRNA^*] &= \frac{k_m}{\gamma_m}, \text{ and} \\ [protein^*] &= \frac{k_m k_p}{\gamma_m \gamma_p}.\end{aligned}\tag{4.2}$$

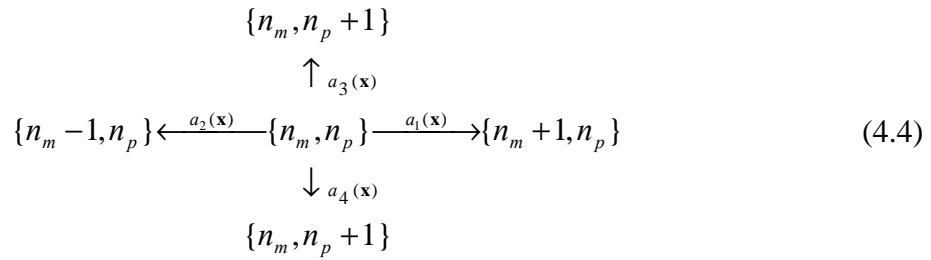
If we consider the particle numbers, rather than concentration for a given volume Ω , the particle numbers of mRNA and protein at steady state can be described by

$$\begin{aligned}n_m^* &= \Omega \frac{k_m}{\gamma_m}, \text{ and} \\ n_p^* &= \Omega \frac{k_m k_p}{\gamma_m \gamma_p},\end{aligned}\tag{4.3}$$

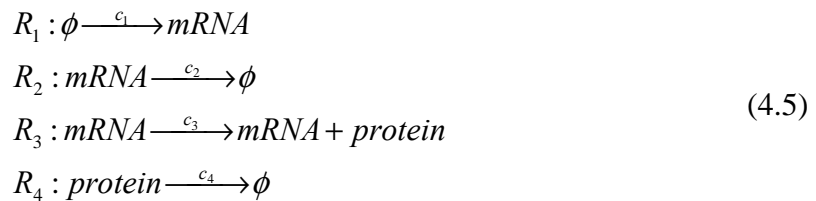
where $n_{m,p}^*$ denotes the number of mRNA and protein at steady state, respectively.

4.1 Numerical simulation result: Gillespie algorithm

Since both the transcription of mRNAs from genes and their subsequent translation into proteins are stochastic biochemical events, the corresponding stochastic model of Eq. (4.1) can be described as the following random birth-and-death Markov process



Here, $a_i(\mathbf{x})$ ($i=1, \dots, 4$) represents the propensity function of the reaction channel R_j and \mathbf{x} denotes the current state or $\begin{bmatrix} n_m \\ n_p \end{bmatrix}$, where $n_{m,p}$ denotes the number of mRNA and protein, respectively. To simulate this stochastic process in Gillespie algorithm, the reaction channels can be described as following



Here, the reaction channel R_1 describes the event that “one” mRNA is produced (from DNA transcription) with the state change vector $\mathbf{v}_1 = \begin{bmatrix} +1 \\ 0 \end{bmatrix}$, where v_{11} represents the change in mRNA, and v_{12} represents the change in protein. The reaction

channel R_2 describes the event that “one” mRNA molecule is degraded, with the state change vector $\mathbf{v}_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$. Same descriptions can be made for R_3, R_4 . Each c_j represents the specific probability rate constant of reaction channel R_j , as discussed in chapter 3.1. Suppose the system’s volume Ω is given, the value of c_j can be derived from the corresponding rate constant in Eq. (4.1) according to Eq. (3.8)

$$\begin{aligned}
 c_1 &= \Omega k_m \\
 c_2 &= \gamma_m \\
 c_3 &= k_p \\
 c_4 &= \gamma_p
 \end{aligned} \tag{4.6}$$

For simulation, we refer to [25], [26] and choose biologically relevant parameter set for bacteria, where

Table 4-1 Reaction rate constants for single gene expression

Parameter	value	unit
Ω	1.7	fl(10^{-15} L)
k_m	0.002	$M s^{-1}$
γ_m	0.01	s^{-1}
k_p	0.2	s^{-1}
γ_p	0.0004	s^{-1}

We have assumed that $\gamma_m \gg \gamma_p$, since mRNA typically degrades faster and its lifetime is much shorter than that of a protein. The system's volume Ω represents cell's volume. According to Eq. (4.2), the concentration of mRNA at steady state is 0.2nM. In other words, the particle number of mRNA at steady state is 0.2 for each cell. It means on average, one mRNA of this gene can be found for every five cells. Besides, the concentration of protein at steady state is 100nM. If we consider Ω equals 1.7fL, the particle number of protein at steady state is 100 for each cell. For stochastic modeling, we use probability rate constant instead of reaction rate constant. According to Eq. (3.8), the corresponding probability rate constants can be calculated as shown in the next table

Table 4-2 Probability rate constants for single gene expression

Parameter	value	unit
c_1	0.002	s^{-1}
c_2	0.01	s^{-1}
c_3	0.2	s^{-1}
c_4	0.0004	s^{-1}

Using parameters mentioned above, the simulation results of Gillespie algorithm are demonstrated as following figures. Dashed line represents the analytical solution of protein in deterministic kinetic O.D.E. (Eq. (4.2)).

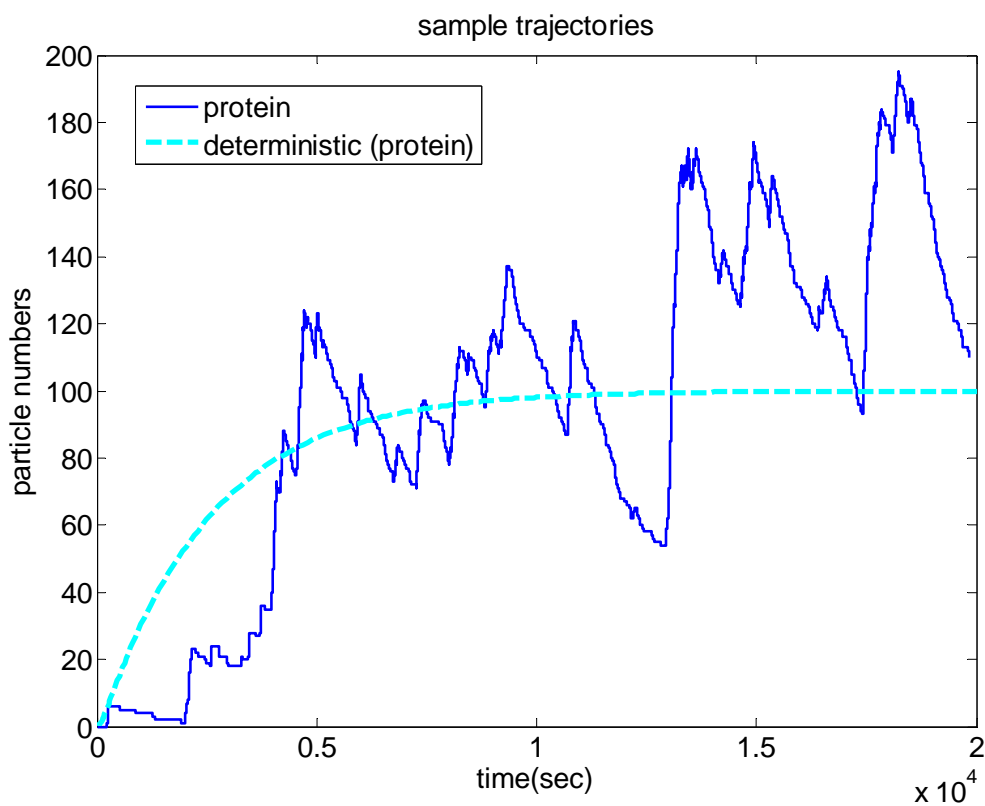
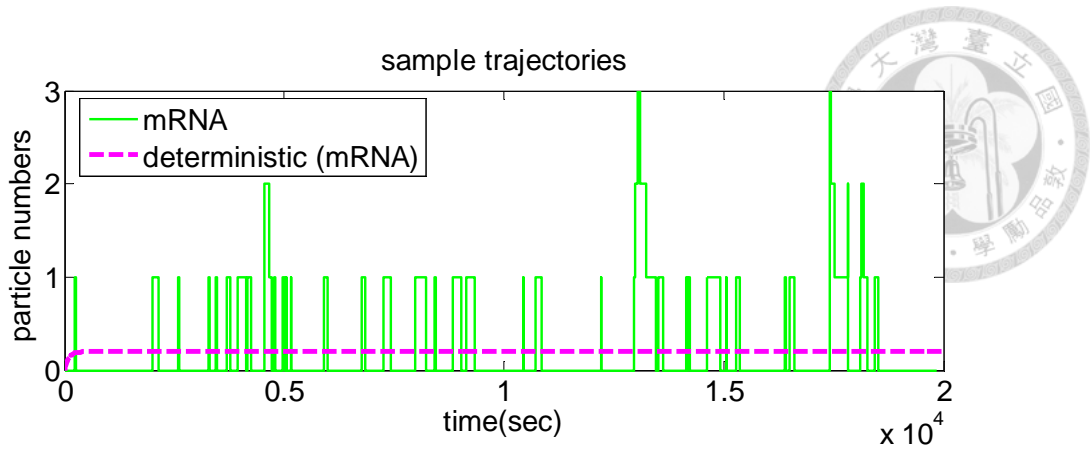


Figure 4-2 A sample stochastic trajectory of mRNA (top) and protein (bottom). Shown are stochastic trajectories simulated by Gillespie algorithm with parameters listed in Table 4-1.

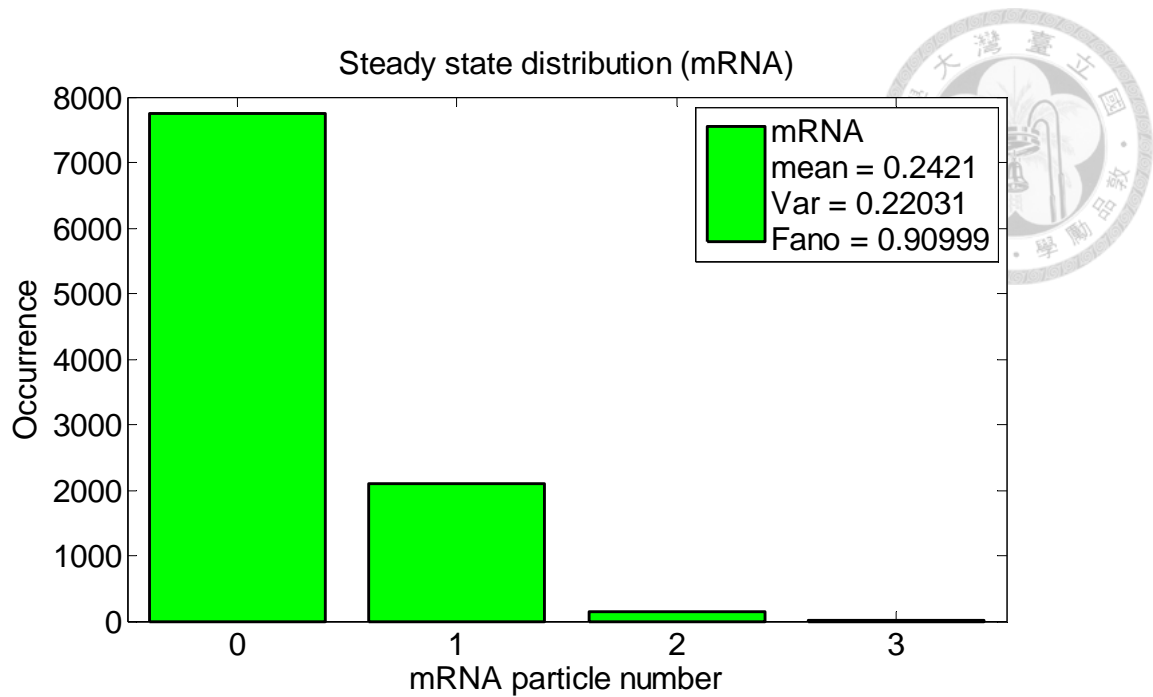


Figure 4-3 Steady state distribution of mRNA in single expression model.

Shown are the mRNA particle distribution in 10,000 samples obtained by the Gillespie algorithm.

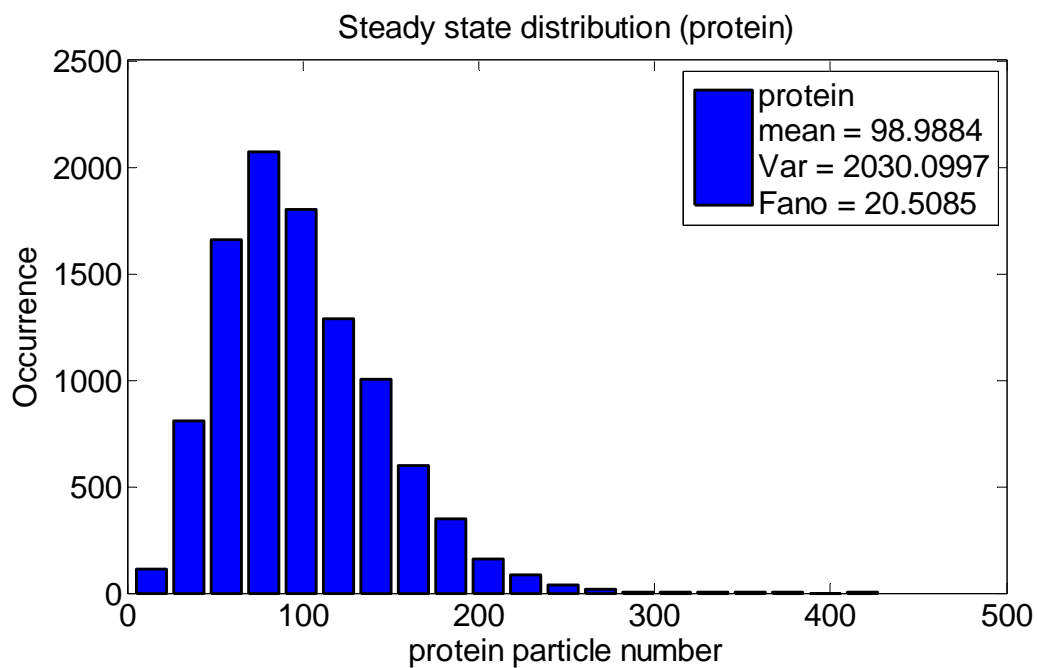


Figure 4-4 Steady state distribution of protein in single expression model.

Shown are the protein particle distribution in 10,000 samples obtained by the Gillespie algorithm.

To analyze the steady state distribution, we use Fano factor which is defined as variance/mean to measure the noise.

In this model, the analytical Fano factor for mRNA is 1.00 [25], means a Poisson distribution, and the simulation result of mRNA Fano factor in Figure 4-3 is about 0.91. According to Ref. [27], the analytical Fano factor for protein can be derived by *linear noise approximation*, and the analytical Fano factor of steady state protein distribution will be:

$$Fano_{(protein)} = \frac{(\sigma_p^*)^2}{\langle n_p^* \rangle} \approx 1 + \frac{\langle n_p^* \rangle}{\langle n_m^* \rangle} \frac{\gamma_p}{\gamma_m + \gamma_p}. \quad (4.7)$$

Since mRNA's life time is much shorter than protein's in real biological system, the value of degradation rate γ_m is actually much larger than γ_p (see Table 4-1).

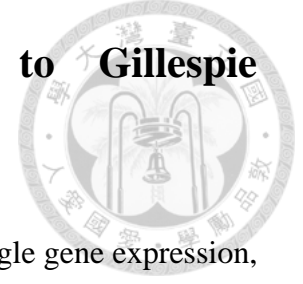
Together with the fact that $\langle n_m^* \rangle = n_m^* = \Omega \frac{k_m}{\gamma_m}$ and $\langle n_p^* \rangle = n_p^* = \Omega \frac{k_m k_p}{\gamma_m \gamma_p}$, the Eq. (4.7)

becomes

$$Fano_{(protein)} \approx 1 + \frac{k_p}{\gamma_m}. \quad (4.8)$$

Where $\frac{k_p}{\gamma_m}$ represents the average number of protein produced per mRNA, and can be regarded as translation efficiency. This analytical result also shows that most of noise in central dogma comes from translational level, which has been discussed in Ref. [25] and demonstrated with experiments [28]. The analytical Fano factor of protein is about 21.00 in this case, and is 20.51 for simulation result in Figure 4-4, consistent with the analytical result.

4.2 Failure in Langevin approximation to Gillespie algorithm in single gene expression model



In order to dissect the noise out of this stochastic process in single gene expression, we rewrite the scheme in Eq. (4.4) into Langevin form. According to Eq. (3.37), the corresponding Langevin form is

$$\begin{aligned}
 n_m(t+dt) &= n_m(t) \\
 &+ [c_1 dt + \sqrt{c_1 dt} N_1(0,1)] \\
 &- [c_2 n_m(t) dt + \sqrt{c_2 n_m(t) dt} N_2(0,1)], \text{ and} \\
 n_p(t+dt) &= n_p(t) \\
 &+ [c_3 n_m(t) dt + \sqrt{c_3 n_m(t) dt} N_3(0,1)] \\
 &- [c_4 n_p(t) dt + \sqrt{c_4 n_p(t) dt} N_4(0,1)].
 \end{aligned}
 \tag{4.9}$$

For particle number of mRNA n_m , the second term inside the bracket describe the average increment $c_1 dt$ and the corresponding fluctuation $\sqrt{c_1 dt} N_1(0,1)$ for reaction channel R_1 , which is the constant production from DNA to mRNA (Eq. (4.5)). Similarly, the third term inside the bracket describes the mRNA degradation channel R_2 . $N_1(0,1)$ and $N_2(0,1)$ are mutually independent unit normal random variable. Similarly, for particle number of protein n_p , the second term describe the translation of mRNA into protein, and the third term describe protein degradation. Since dt must satisfies the two conditions discussed in section 3.3.2 to make Eq. (4.9) a good approximation, and it will be valid especially for the system with large number of molecules. Unfortunately, in real biological system, the copy number of mRNA is usually too low to make a good approximation. In the following figures, we simulate the two-step gene expression model with three different sizes of dt , using the same parameters in Table 4-1, and compare to the result of Gillespie algorithm:

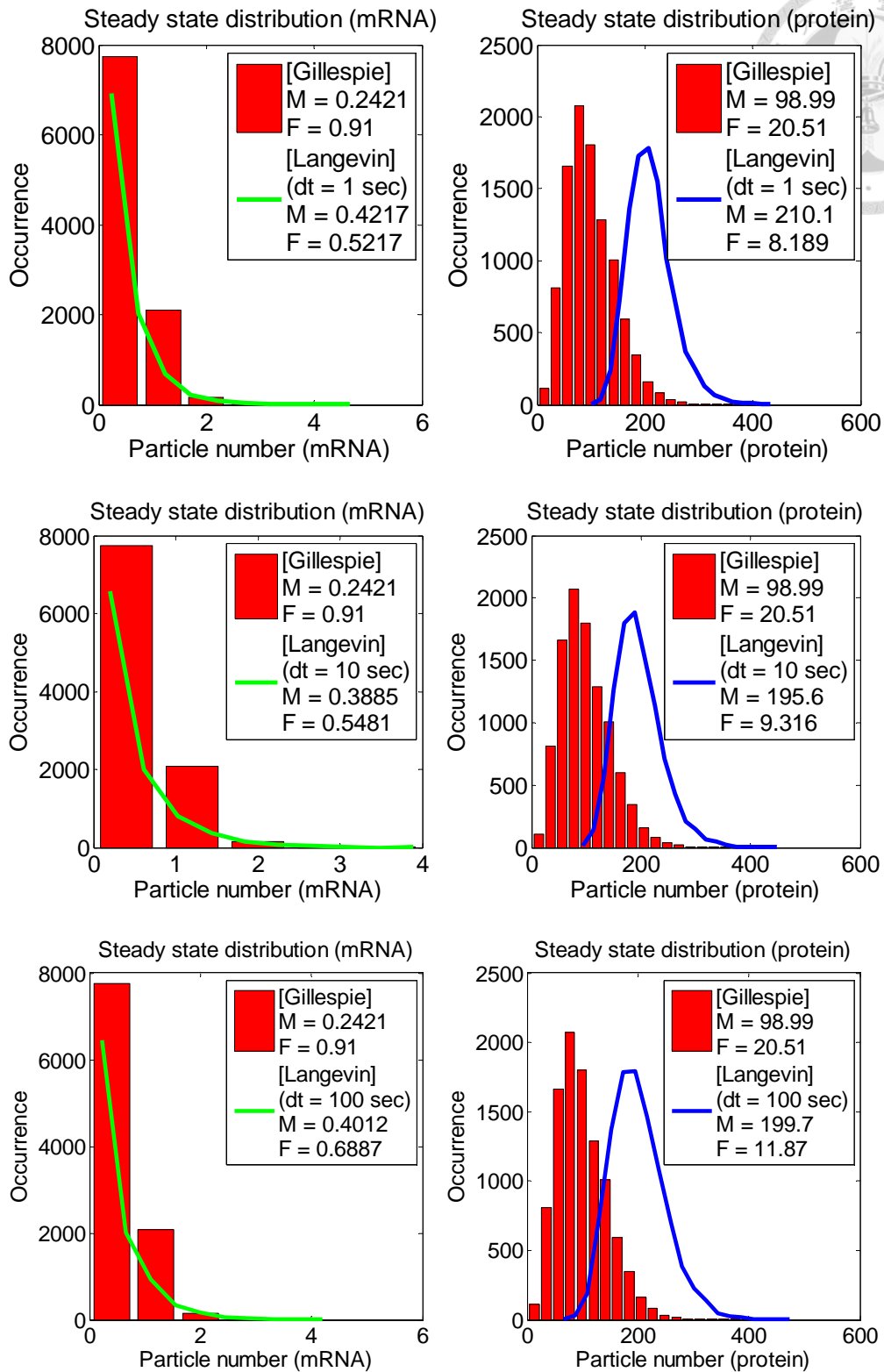


Figure 4-5 Approximating Gillespie algorithm with different size of dt in Langevin. (M denotes “mean” and F denotes “Fano factor,” respectively). Parameters used in simulation are listed in Table 4-1 of section 4.1.

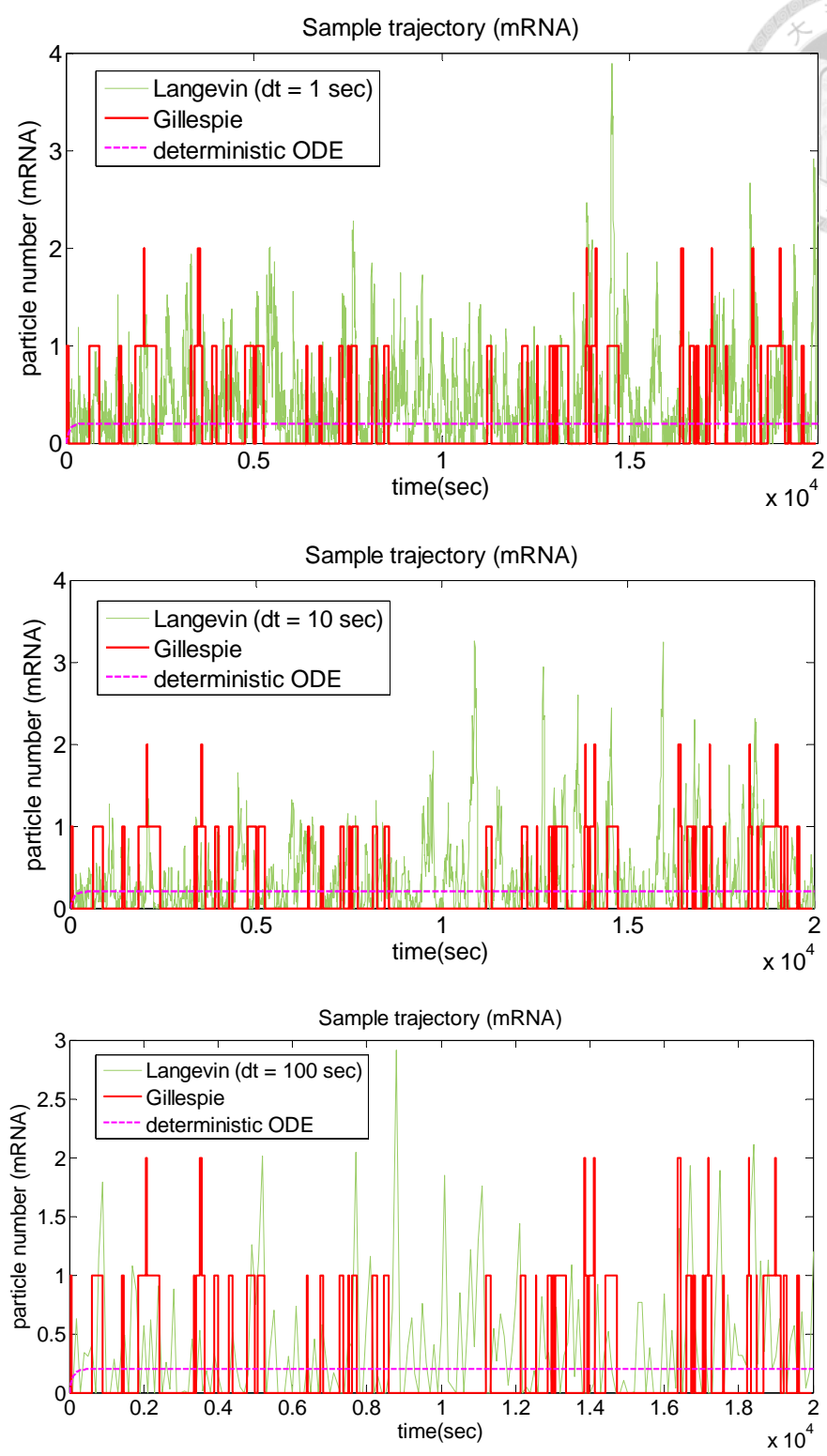


Figure 4-6 Langevin trajectory for mRNA (green) with different size of dt , compared to the stochastic trajectory of Gillespie (red) and deterministic ODE solution (dashed line). Parameters used in simulation are listed in Table 4-1 of section 4.1.

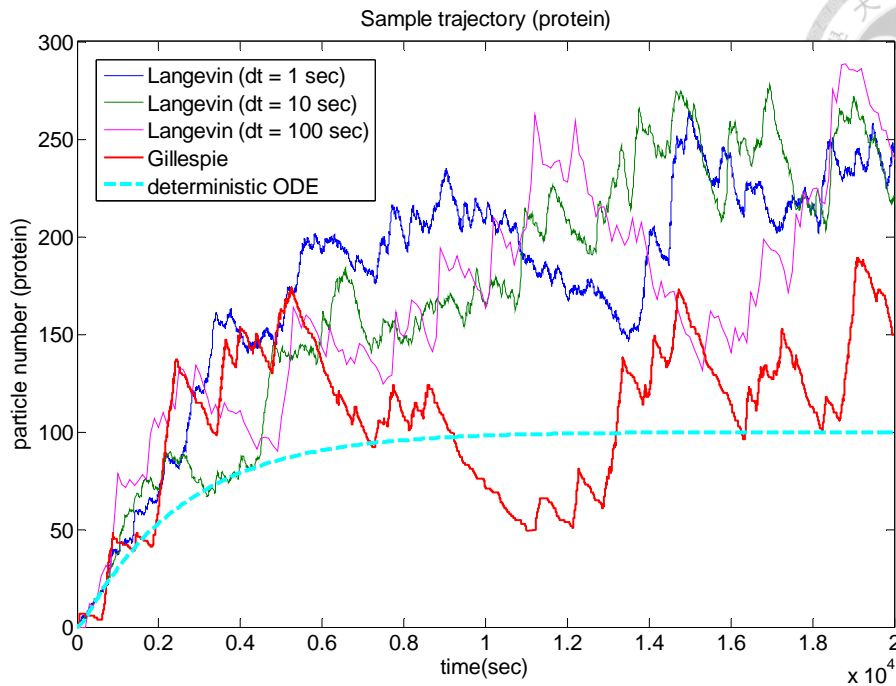
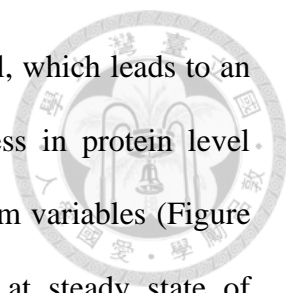


Figure 4-7 Langevin trajectories for protein with different size of dt , compared to the trajectory of Gillespie algorithm and deterministic ODE solution. Parameters used in simulation are listed in Table 4-1 of section 4.1.

The results of Figure 4-5 and Figure 4-6 show that most of errors result from the failure in approximating Poisson random variables with normal random variables at mRNA level. At steady state, the copy number of mRNA is small and most of cell contains no mRNA (0.2 at steady state of deterministic ODE as shown in Eq. (4.3)). Under this situation, using a continuous normal random variable to approximate these rare events is obviously unwise. While stochasticity in Langevin is generated by the normal distributed noise term which takes value from a continuous real number space (Eq. (3.37)), the Gillespie algorithm is a discrete particle number based simulation. Hence, it is inevitable that when Gillespie algorithm takes zero value but the Langevin is a small nonzero real number. As shown in Figure 4-6. This invalid approximation for rare event case eventually leads to a higher steady state average of mRNA in Langevin.



The error in mRNA level will pass to downstream protein level, which leads to an invalid approximation of protein distribution while the discreteness in protein level seems to be fine when approximated with continuous normal random variables (Figure 4-7). This is due to higher quantity in protein molecules (100 at steady state of deterministic ODE as shown in Eq. (4.3)). As we have discussed in section 3.3.2, higher particle number can lead to higher propensity function, which makes it easier to find a proper dt that is large enough for a Poisson random variable to be approximated by a normal random variable (Eq. (3.33)).

Although the approximation could be better if we raise the steady state level of mRNA, but it will be kind of contradictory to the simulation of noise behavior in biology since the noise is less important in a higher copy number system, not to mention mRNA copy number is actually very low in nature. Another way to solve the problem of discreteness in a rare event case is using a larger dt to compensate a small propensity function, but a large dt will be susceptible to the change of system state, and it will be against the leaping condition (Table 3-3), which requires dt to be small enough to make every reaction occurs within dt not to change the propensity function significantly. In order to inspect the combinational effect of higher copy number and larger dt , we scan for the parameters set with different sizes of k_m and different sizes of dt in corresponding Langevin equation while fixing the other parameters. For each combination (k_m, dt) , the steady state values of 10,000 Langevin trajectories are collected and the statistics are computed to give the mean and Fano factors. Error rate is defined as $\left(\frac{\text{simulated result}}{\text{analytical result}}\right) \times 100\% - 1$. The result is demonstrated in the following figures:

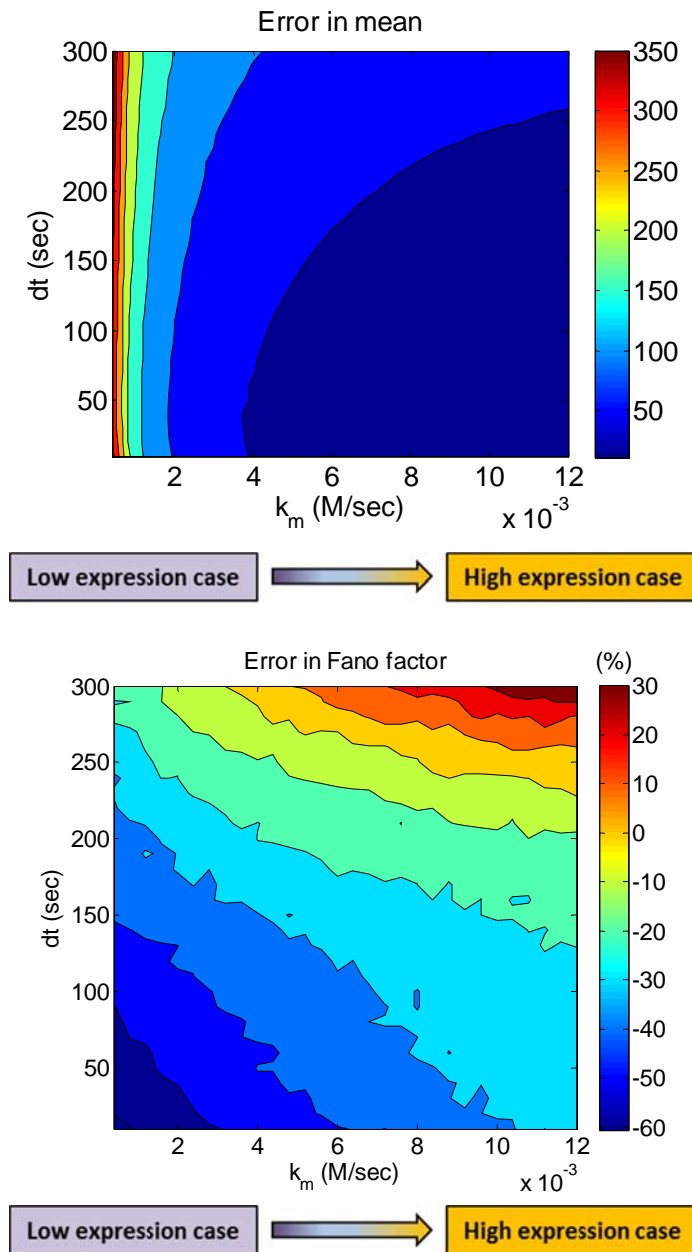


Figure 4-8 The errors of Langevin simulation as compared to analytical results, different combination of parameters (k_m, dt) is used for scanning. Shown are error in Mean (upper panels) and in Fano factor (lower panels) for traditional Langevin (Eq. (4.9)). The error are the difference between the statistic values of proteins from 10,000 trajectories as compared to analytical results at steady state (Eq. (4.8) in section 4.1).



The scanning result of Figure 4-8 shows different error levels under different (k_m, dt) . We found that although there exists some “safe region” (yellow strip) in which the error of Fano factor is around 0%, this Langevin form is still inapplicable if we consider a model of gene regulation where the k_m of a downstream gene is regulated and the value changes as time progresses. This means if we want to simulate a model of gene regulation with Langevin equation, the time step dt needs to be “reset” each time the system proceed a unit time of dt in order to keep a well approximation to Gillespie algorithm. Furthermore, even if the Fano factor is right, it might be the consequence of wrong mean and average. For instance, if we take a look at $(k_m = 6 \times 10^{-3} \text{ M/s}, dt = 250s)$ in yellow region, the actual Langevin simulation result is the following

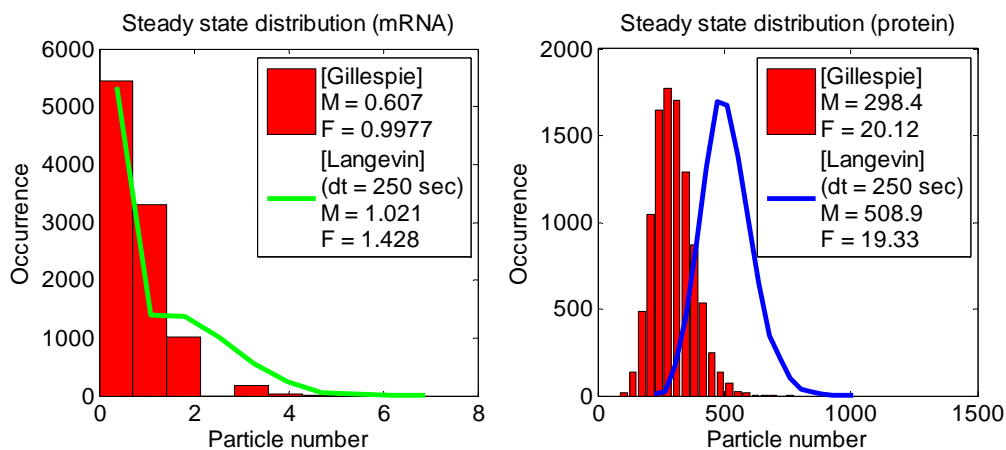


Figure 4-9 Approximating Gillespie algorithm with $(k_m = 0.006, dt = 250)$ in Langevin equation. (M denotes “mean” and F denotes “Fano factor,” respectively)

Since even k_m increase to $6 \times 10^{-3} \text{ M/s}$ compared to original $2 \times 10^{-3} \text{ M/s}$ in Table 4-1, the analytical steady state copy number of mRNA is just 0.6 and it is still a

rare event case. Nevertheless, it is still possible to apply this Langevin approximation on different parameter set. According to single molecular level counting of mRNA in living cell [29], the number of mRNA molecular of MS2 coat protein in *E. coli* can be 5~10 which is 25~50 times of original parameters in Table 4-1 at steady state. In this case, a proper dt can be found for Langevin simulation. The next figure shows a valid approximation with parameters ($k_m = 0.08$, $dt = 50$), which is the case for high expression rate.

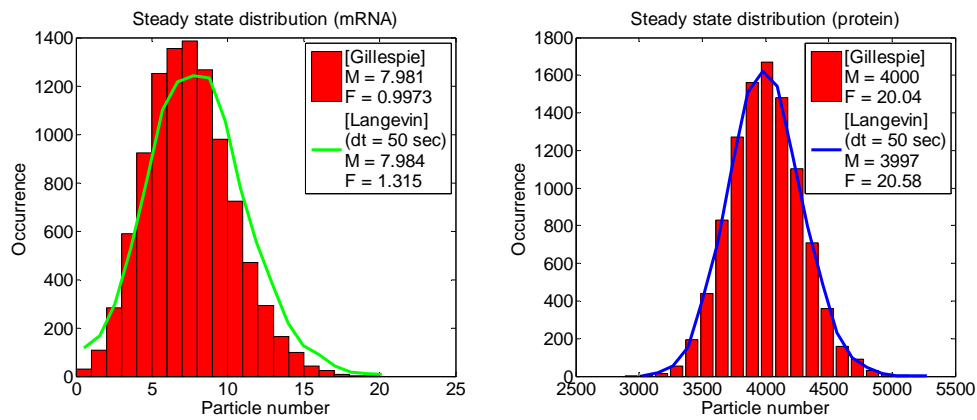


Figure 4-10 Particle number distribution in mRNA (left) and Protein (right) at a higher expression rate with parameters ($k_m = 0.08$, $dt = 50$), the figure shows that Langevin approximation can be valid in a higher expression case.

In general, while different gene has different level of steady state mRNA molecules, the copy number of mRNA is still low for most cases. Besides, each gene has “ON” and “OFF” states. For gene in “OFF” state, the expression level will definitely lower than ON state hence has much fewer mRNA molecules. In simulation aspects, low copy number of mRNA is the main reason that Langevin equation fail to approximate Gillespie algorithm, and it is not practical to use an approach that we are not sure when it is valid. Hence, we need to find another approach to dissect the noise out of Gillespie algorithm, and this approach must be valid for most cases.

Chapter 5 The Burst Production Model of Single Gene Expression



Recent studies have shown the burst-like phenomenon in protein production for a given gene [25], [26], [29], and it is demonstrated by real-time measurements on individual *E. coli* cell in single molecule level [30], [31]. This bursting is a suddenly rise of protein particle number, which can be regarded as many single translational event happens within the observed time scale. Each single production event must be really fast, and fast enough to accumulate before the particle number drops down by degradation or cell division. When a burst production occurs with a different burst size, it generates strong fluctuation and results in steady state noise. According to Ref. [30], the experimentally observed translational burst size distribution can be well fitted by an exponential distribution.

For translational bursting, the expression of proteins from a given gene can be characterized by two parameters: the average frequency of burst events per cell cycle α , and the average number of protein molecules produced per burst β [26][30], both parameters can be measured experimentally [30].

The analytical distribution of burst events can be derived from simple two-step model of gene expression (Figure 4-1) [26]. Since the life time of mRNA is much shorter than the life time of protein (which means $\gamma_m \gg \gamma_p$, similar to the example parameters in Table 4-1), as in the case of bacteria or yeast, the fluctuation in mRNA level can be neglected and proteins can be considered to be produced in uncorrelated random events, with the frequency equal to mRNA production rate (or k_m in Figure 4-1). Besides, although γ_p describes the rate of decrease in protein molecules, it is

actually composed of protein degradation and protein dilution after cell division. For long half-life time case such as fluorescent reporters, γ_p is mostly due to protein dilution rate. Hence the experimental observed average frequency of burst events per cell cycle can thus be described by

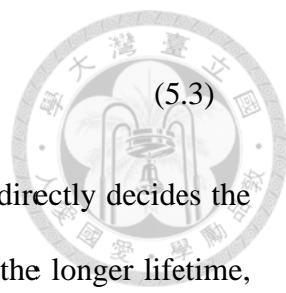
$$\alpha = \Omega \frac{k_m}{\gamma_p}, \quad (5.1)$$

where Ω represents system's volume. Besides, the average number of protein molecules produced per burst β can be regarded as the average number of protein translated from an mRNA molecule before it is degraded, and it can be written as

$$\beta = \frac{k_p}{\gamma_m}. \quad (5.2)$$

The result of $\alpha * \beta$ equals $\Omega \frac{k_m k_p}{\gamma_m \gamma_p}$, which is consistent with Eq. (4.3) and

will be the average protein molecules of each cell. Owing to the stochastic nature, each mRNA can produce different copy numbers of protein, which is the consequence of different mRNA lifetime. In two-step model gene expression model, the degradation of mRNA is an exponential decay with the average lifetime $\frac{1}{\gamma_m}$, and the lifetime distribution of mRNA molecule is exponentially distributed with the probability density function:



$$f(t; \frac{1}{\gamma_m}) = \gamma_m \exp(-t\gamma_m). \tag{5.3}$$

The size of mRNA lifetime t (which is a random variable) indirectly decides the size of burst contributed by each mRNA molecule. In other words, the longer lifetime, the more protein can be translated. Assume each translational bursting event is the consequence of fast translation by an mRNA molecule before its degradation, since the transcription rate is k_p , and the random variable of observed burst size x can be described as tk_p , with the probability density function

$$f(x; \beta) = \frac{1}{\beta} \exp(\frac{-x}{\beta}), \tag{5.4}$$

where $\beta = \frac{k_p}{\gamma_m}$ as denoted in Eq. (5.2). Hence, each burst size can be described by an exponential random number with mean equals to β . This exponential burst size distribution is consistent with the experimental observation [30]. The concept of translation bursting is illustrated by the following figure:

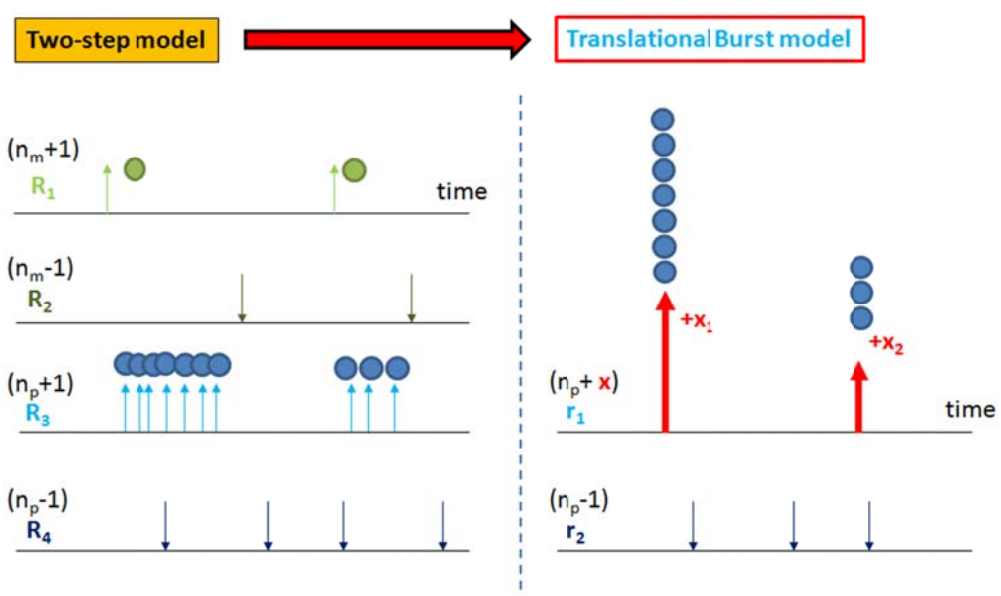


Figure 5-1 Transformation of the two-step gene expression model into the translational burst model.

The notation R_1 to R_4 on the left hand side in Figure 5-1 represents the reaction channels described in Eq. (4.5), which are mRNA production, mRNA degradation, protein production and protein degradation, respectively. An mRNA molecule can be translated into some protein molecules before its degradation, and the amount of protein that translated by one mRNA molecule (burst size) is a random variable, denoted as x on right hand side of Figure 5-1, and it is experimentally observed (see Ref. [30]) to be exponentially distributed, with the probability density function described in Eq. (5.4). The notation r_1 on the right hand side represents the translational burst channel, and it can be regarded as a simplified expression of R_1 to R_3 . Notation r_2 represents first order degradation channel of protein, same as R_4 in two-step model.

For translational burst model, the corresponding master equation at steady state can be solved analytically [26], and the protein distribution at steady state will be a Gamma distribution, with shape parameter $\alpha = \Omega \frac{k_m}{\gamma_p}$, scale parameter $\beta = \frac{k_p}{\gamma_m}$, and the corresponding probability density function $g(x)$:

$$g(x; \alpha, \beta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} \exp\left(-\frac{x}{\beta}\right) \quad (\alpha, \beta > 0 \text{ and } \alpha, \beta \in \mathfrak{R}). \quad (5.5)$$

The mathematical form of Eq. (5.5) is exactly the probability density function of a Gamma distribution, with Γ denoting the standard Gamma function, or

$$\Gamma(n) = \int_0^{\infty} e^{-u} u^{n-1} du \quad (5.6)$$

Both α and β are positive real numbers. If we consider the parameter set in Table 4-1, the value of α equals 5, and the value of β equals 20. The next table shows the Gillespie simulation result (which has been demonstrated in Figure 4-4) and the corresponding Gamma distribution with the same parameters.

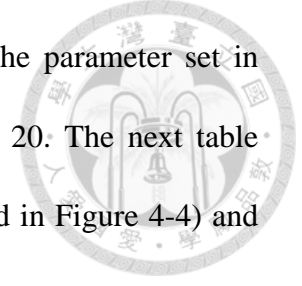


Table 5-1 Statistical quantities derived from Gillespie simulation and analytical Gamma distribution.

Statistics	Statistics of $Gamma(\alpha, \beta)$	Statistics of $Gamma(5,20)$	Gillespie algorithm in two-step model simulation
Mean	$\alpha\beta$	100.00	98.99
Variance	$\alpha\beta^2$	2000.00	2030.10
Fano factor	β	20.00	20.51

The Fano factor of $Gamma(\alpha, \beta)$ equals to β , which is close to $1 + \beta$ derived from linear noise approximation in (see Eq. (4.8)). The result of Gamma distribution captures the asymmetry of experimentally observed distributions in Refs. [28], [32], [33]. We notice that while the derivation of Gamma distribution does include an additional explicit description of exponential distribution in burst size, but the result is consistent with Gillespie simulation on two-step model of single gene expression (see Figure 4-1). It is because the exponential burst distribution is inherently described in original two-step model through exponential decay of mRNA molecules [34].

5.1 Modified Gillespie algorithm in burst production model

Since the two-step model and burst model give the same result, we can now restate the two-step model in Figure 4-1 into translational burst model:

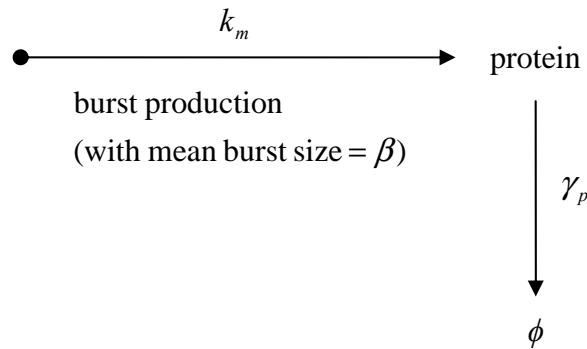


Figure 5-2 Translational burst production model for central dogma.

This translational burst production model is the consequence of two-step model in Figure 4-1, and the only difference is that the “burst” requires the rate of translation to be fast enough to make burst occur. In other words, each mRNA molecule is able to be translated into few proteins before its degradation. In Figure 5-2, γ_p denotes the rate of protein degradation, and k_m denotes the rate of mRNA production. Both k_m and γ_p are the same as the symbols used in Figure 4-1. Since the exponential distribution of burst size comes from the exponential distribution of mRNA life time, it is automatically assumed that each mRNA molecule *only* produces “one burst.” Hence, the rate of mRNA production can also be regarded as the burst rate. Besides, we do not worry about the possibility that an mRNA molecule is degraded without being translated, since it is already included in exponential distribution and this probability is equal to the probability when burst size is zero.

The burst production model in Figure 5-2 can be simulated with modified Gillespie algorithm, and the reaction channels will be:



Here, x is an exponential random variable with mean β , which has the probability density function described in Eq. (5.4). The reaction channel r_1 describes the burst production of protein with state change vector $\mathbf{v}_1 = [+x]$, and r_2 describes protein degradation with state change vector $\mathbf{v}_2 = [-1]$. In conventional Gillespie algorithm, each component v_{ji} (see Eq. (3.3)) in \mathbf{v}_j is a fixed integer, usually +1 for production and -1 for degradation. For burst model, the state change vector of burst production channel (reaction channel R_1 in this case) contains a random variable to describe a random burst, and it is an exponential random number with mean equals to β as discussed in Eq. (5.4).

Since Gillespie algorithm is a discrete number based simulation, we can take an exponential random number and round it off each time when burst channel is chosen. The following figures demonstrate the simulation result of modified Gillespie algorithm in burst production model.

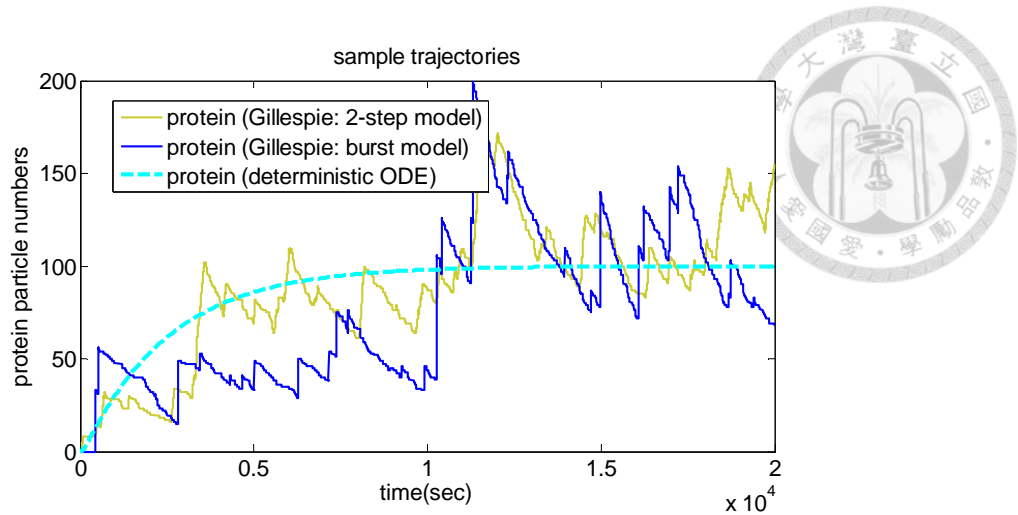


Figure 5-3 Sample trajectories of Gillespie in two-step model and burst model. The parameters used in simulation are listed in Table 4-1 of section 4.1.

The next figure shows the steady state distribution for 10,000 trajectories. The results are quite similar for two-step model and burst model with the same parameter set in Table 4-1. The mRNA is not shown in burst model, since the mRNA production rate is now burst rate, and the mRNA degradation rate is involved in burst size description.

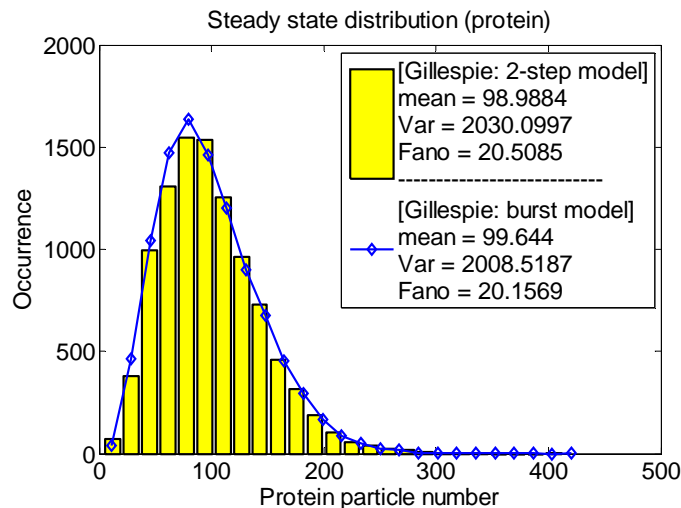
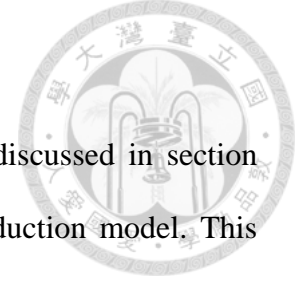


Figure 5-4 Steady state distributions of protein in two-step model and in burst model. Shown are distributions from 10,000 trajectories simulated with the parameters listed in Table 4-1 of section 4.1. The simulation results of both models in Gillespie algorithm are quite the same.

5.2 The Langevin form of burst production



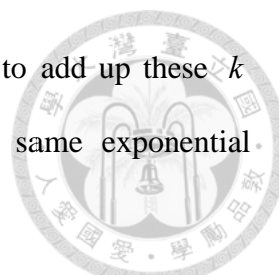
In this section, we use the same argument which has been discussed in section 3.3.1 and section 3.3.2 to derive the Langevin form of burst production model. This argument requires two specific conditions to be satisfied so that the Gillespie algorithm can be approximated by Langevin equation. In general, if we can find a proper Langevin dt that satisfies the two conditions, then it will be valid to use Langevin to approximate Gillespie algorithm as discussed in section 3.3, and same argument can be made for the Gillespie algorithm in burst model. We found that for simulating a single gene expression model, the simplified burst production model can “bypass” the rare event problem of mRNA by directly describing the stochasticity in protein level. For simulation aspect, if we consider merely the protein noise, this “burst-Langevin” will be a better choice than the original Langevin in Ref. [19] to approximate the stochastic process of two-step single gene expression model.

Before we begin to derive this “burst-Langevin” equation, we first introduce a distribution called *Erlang distribution*.

5.2.1 The Erlang distribution

Suppose at the ticket office of a theater, there are k people waiting in front of someone (see Figure 5-5), k is of course a positive integer. If we assume that the amount of time each people spend at the ticket office is a random variable exponentially distributed with mean β , which has the probability density function

$$f(x; \beta) = \frac{1}{\beta} \exp\left(\frac{-x}{\beta}\right). \quad (5.8)$$



So how long does one have to wait? An intuitive thinking is to add up these k “independent” exponential random variables, which are from a same exponential distribution with mean β .

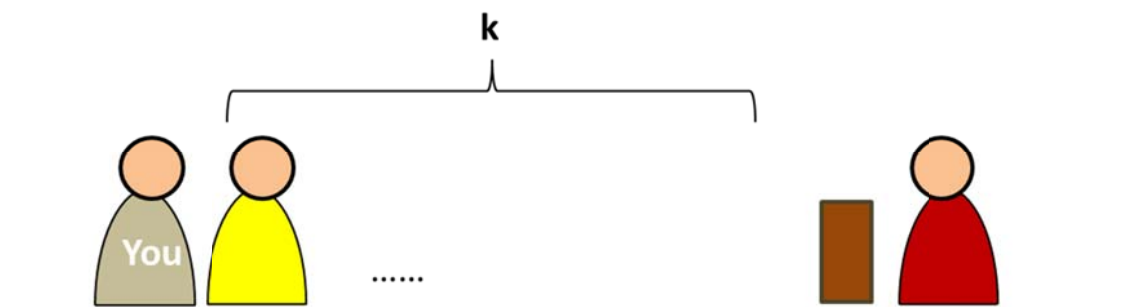


Figure 5-5 Illustrated queuing problem

Hence, your waiting time is a random variable, and the distribution is simply summing up k independent exponential random variables with mean β , or

$$x = \sum_{i=1}^k \text{Exponential}_i(\beta), \quad (5.9)$$

where $\text{Exponential}_i(\beta)$ denotes the i th independent exponentially distributed random variable, and the distribution of waiting time x can be described by the probability density function:

$$f(x; k, \beta) = \frac{1}{\beta^k \Gamma(k)} x^{k-1} \exp\left(-\frac{x}{\beta}\right) \quad (k, \beta > 0 \text{ and } k \in \mathfrak{N}, \beta \in \mathfrak{R}). \quad (5.10)$$

This is the Erlang distribution, which has two parameters. The notation k in Eq. (5.10) is also called the *shape parameter*, and the notation β is also called the *scale parameter*. Actually, the Erlang distribution is just a special case of Gamma distribution, where the latter does not restrict k to be a positive integer (see Eq. (5.5) for probability density function of Gamma distribution).

5.2.2 Derivation of the Langevin form of burst production

Since the random process can be exactly simulated by Gillespie algorithm, either for two-step model or burst model, and we have shown that for single gene expression, the two-step model can be simplified into burst model in section 5.1. The advantage of burst model is that: the description of mRNA is implicit. It helps to bypass the problem of invalid Langevin approximation to two-step model in low gene expression case. In this section, we use the Langevin equation to approximate the burst model, instead of the two-step model, to describe the stochastic process of a single gene expression.

The concept of approximating Gillespie algorithm with a Langevin equation is very similar to what has been discussed in section 3.3. Here, we use the same notations as in Chapter 3 that describe the Gillespie algorithm. Consider a system with size Ω contains N different reactant species S_i ($i=1,\dots,M$) and M different reaction channels R_j ($j=1,\dots,M$). Suppose the system's state $\mathbf{X}(t)$ at the current time t is denoted as \mathbf{x}_t . Let $K_j(\mathbf{x}_t, \tau)$ ($\tau > 0$) be the firing times of reaction channel R_j ($j=1,\dots,M$) within time step τ , as previously defined in Eq. (3.27). Given that the reactant species S_p ($p \in [1, N]$) is produced in a burst-like production with average burst size β through a burst production channel R_b ($b \in [1, M]$) with the state change vector \mathbf{v}_b , which has an exponential random variable $Exponential(\beta)$ as the p th component, or simply $v_{bp} = Exponential(\beta)$. The term *Exponential* denotes an exponentially distributed random variable, and $Exponential(\beta)$ is an exponentially distributed random variable with mean equals to β , and the probability density function is just the same as that in Eq. (5.4). For the reactant species which are not produced in burst, the particle numbers at time $t + \tau$ can be described as:

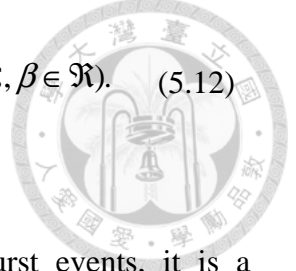
$$X_i(t + \tau) = X_i(t) + \sum_{j=1}^M K_j(\mathbf{x}_t, \tau) \nu_{ji} \quad (i = 1, \dots, N; i \neq p), \quad (5.11)$$

where X_i denotes the particle number of species S_i . The description of Eq. (5.11) is just as same as that has been discussed in section 3.3.1.

For the reactant species S_p that is produced in burst, the increment of particle number within time step τ from burst production channel, can be described by two random variables: the random number of burst events within τ , denoted as $K_b(\mathbf{x}_t, \tau)$, and the random burst size of each burst event. Suppose the burst is “independent” of each other, we can simply sum up all these independent burst to get the net production of burst channel within time step τ . Since we consider the burst size distributed exponentially with mean β , the net production of burst channel is to add up these independent exponentially distributed random variables $Exponential(\beta)$, and this is just like the case of Erlang distribution that we have discussed in section 5.2.2. The only difference is that: the number of random variables $Exponential(\beta)$ is another random variable $K_b(\mathbf{x}_t, \tau)$. In other words, for the example of buying ticket in the movie theater, as we just discussed in section 5.2.2, this time we don’t know how many people there are in front of us.

Here, we use the notation K_b to replace $K_b(\mathbf{x}_t, \tau)$ for simplification. The net production of burst channel within time step τ can be described by an Erlang random variable with a random shape parameter K_b , and a scale parameter β . The result can be denoted as $Erlang(K_b, \beta)$. The term *Erlang* denotes an Erlang random variable, which has the probability density function:

$$f(x; K_b, \beta) = \frac{1}{\beta^{K_b} \Gamma(K_b)} x^{K_b-1} \exp\left(-\frac{x}{\beta}\right) \quad (K_b, \beta > 0 \text{ and } K_b \in \mathfrak{N}, \beta \in \mathfrak{R}). \quad (5.12)$$



Since the random variable K_b describes the number of burst events, it is a positive integer, the random variable $Erlang(K_b, \beta)$ still lies in the domain of Erlang distribution.

Hence, for the reactant species S_p that is produced in burst, the particle number of S_p at time $t + \tau$ is the net effect of non-burst channels, plus the net production of burst channel within τ , and it can be described as:

$$X_p(t + \tau) = X_p(t) + \sum_{j=1; j \neq b}^M K_j(\mathbf{x}_t, \tau) v_{jp} + Erlang(K_b, \beta). \quad (5.13)$$

Here, the notation K_b denotes $K_b(\mathbf{x}_t, \tau)$, which represents total number of burst events occur within τ , and the notation β is a real number, representing the mean of exponentially distributed burst size.

Suppose τ satisfies the leaping condition in Table 3-3, which requires τ to be small enough so that the propensity for each reaction channel only change a little as discussed in Eq. (3.29), then we can use a Poisson random variable to “approximate” firing times $K_j(\mathbf{x}_t, \tau)$ for every reaction channel (including burst production channel in our case). For the same reason, the number of burst events K_b can be approximated by a Poisson random variable, and the Eq. (5.13) becomes:



$$\begin{aligned}
X_p(t + \tau) = X_p(t) + \sum_{j=1; j \neq b}^M \text{Poisson}(a_j(\mathbf{x}_t), \tau) v_{jp} \\
+ \text{Erlang}(\text{Poisson}(a_b(\mathbf{x}_t), \tau), \beta),
\end{aligned} \tag{5.14}$$

where a_j ($j = 1, \dots, M; j \neq b$) is the propensity function for non-burst channel, and a_b ($b \in [1, M]$) is the propensity function for burst channel. The term *Poisson* denotes a Poisson random variable. Since a Poisson random variable $\text{Poisson}(a_b(\mathbf{x}_t), \tau)$ is an integer, the third term $\text{Erlang}(\text{Poisson}(a_b(\mathbf{x}_t), \tau), \beta)$ on the right side in Eq. (5.14) still lies in the domain of Erlang distribution.

Suppose we can further find a subset of these τ that can satisfy another condition: require τ to be large enough so that the Poisson random variables can be approximated by normal random variables with same mean and variance, as we have discussed in section 3.3.2. Under this condition, the Eq. (5.14) becomes

$$\begin{aligned}
X_p(t + \tau) = X_p(t) + \sum_{j=1; j \neq b}^M \text{Normal}(a_j(\mathbf{x}_t)\tau, a_j(\mathbf{x}_t)\tau) v_{jp} \\
+ \text{Gamma}(\text{Normal}(a_b(\mathbf{x}_t)\tau, a_b(\mathbf{x}_t)\tau), \beta).
\end{aligned} \tag{5.15}$$

Since the shape parameter K_b which has been approximated by $\text{Poisson}(a_b(\mathbf{x}_t), \tau)$ is now a real number $\text{Normal}(a_b(\mathbf{x}_t)\tau, a_b(\mathbf{x}_t)\tau)$ rather than an integer, the Erlang random variable in Eq. (5.14) is changed to the Gamma random variable in Eq. (5.15). For simplification, we denote the normal random variable $\text{Normal}(a_b(\mathbf{x}_t)\tau, a_b(\mathbf{x}_t)\tau)$ as \bar{K}_b' and rewrite the Eq. (5.15) into

$$X_p(t + \tau) = X_p(t) + \sum_{j=1, j \neq b}^M \text{Normal}(a_j(\mathbf{x}_t)\tau, a_j(\mathbf{x}_t)\tau)v_{jp} + \text{Gamma}(K_b', \beta). \quad (5.16)$$

The third term on the right side of Eq. (5.16) forms a new probability space, it becomes an Gamma variate with random shape parameter K_b' and a fixed scale parameter β . Our next move is to analyze $\text{Gamma}(K_b', \beta)$. The following derivation is inspired by Ref.[35]. Let δ denotes a standard Gamma random variable $\text{Gamma}(k, \beta)$, where

$$\delta \sim \text{Gamma}(k, \beta). \quad (5.17)$$

The Laplace transform of δ will be

$$L_\delta(s) = E[e^{-s\delta}] = \int_0^\infty e^{-s\delta} \text{Gamma}(x; k, \beta) dx. \quad (5.18)$$

Replacing $\text{Gamma}(x; k, \beta)$ with its explicit form of probability density function in Eq. (5.5), we obtain:

$$\begin{aligned} L_\delta(s) &= \int_0^\infty e^{-s\delta} \text{Gamma}(x; k, \beta) dx \\ &= \int_0^\infty e^{-s\delta} \frac{e^{-\frac{x}{\beta}} x^{k-1}}{\beta^k \Gamma(k)} dx \\ &= \frac{1}{\beta^k \Gamma(k)} \int_0^\infty e^{-x(s+\frac{1}{\beta})} x^{k-1} dx. \end{aligned} \quad (5.19)$$

After integration by parts, we have

$$L_{\delta}(s) = E[e^{-s\delta}] = (1 + s\beta)^{-k}. \quad (5.20)$$



Now, let δ_b to be an Gamma variate with random shape parameter K_b' and a fixed scale parameter β , where

$$\delta_b \sim \text{Gamma}(K_b', \beta). \quad (5.21)$$

The Laplace transform of δ_b will be

$$\begin{aligned} L_{\delta_b}(s) &= E[e^{-s\delta_b}] \\ &= E[E[e^{-s\delta_b} | K_b' = k]]. \end{aligned} \quad (5.22)$$

Plugging the result of Eq. (5.20), we have

$$L_{\delta_b}(s) = E[(1 + s\beta)^{-K_b'}]. \quad (5.23)$$

The expected value of δ_b will be the negative first derivative of $L_{\delta_b}(s)$ evaluated at $s = 0$, where

$$\begin{aligned}
E[\delta_b] &= -\frac{d}{ds} L_{\delta_b}(s) \Big|_{s=0} \\
&= -\frac{d}{ds} E[(1+s\beta)^{-K_b'}] \Big|_{s=0} \\
&= \beta E[K_b'].
\end{aligned}$$



Besides, the expected value of δ_b^2 will be

$$\begin{aligned}
E[\delta_b^2] &= \frac{d^2}{ds^2} L_{\delta_b}(s) \Big|_{s=0} \\
&= \frac{d^2}{ds^2} E[(1+s\beta)^{-K_b'}] \Big|_{s=0} \\
&= \beta^2 E[(K_b')^2] + \beta^2 E[K_b'].
\end{aligned} \tag{5.25}$$

Next, we can calculate the variance of δ_b will the aids of Eq. (5.24) and Eq. (5.25):

$$\begin{aligned}
Var(\delta_b) &= E[\delta_b^2] - E^2[\delta_b] \\
&= (\beta^2 E[(K_b')^2] + \beta^2 E[K_b']) - (\beta E[K_b'])^2 \\
&= \beta^2 (Var(K_b') + E[K_b']).
\end{aligned} \tag{5.26}$$

So far, we already know that the random variable $\text{Gamma}(K_b', \beta)$ has the mean equal to $\beta E[K_b']$, and the variance $\beta^2 (Var(K_b') + E[K_b'])$. In our case, K_b' is a normally distributed random variable $\text{Normal}(a_b(\mathbf{x}_t)\tau, a_b(\mathbf{x}_t)\tau)$, which has the mean and variance equal to $a_b(\mathbf{x}_t)\tau$. Hence, we derive the mean and variance of the random variable $\text{Gamma}(K_b', \beta)$, where

$$E[\text{Gamma}(K_b', \beta)] = \beta a_b(\mathbf{x}_t) \tau. \quad (5.27)$$

$$\text{Var}[\text{Gamma}(K_b', \beta)] = 2\beta^2 a_b(\mathbf{x}_t) \tau. \quad (5.28)$$



Since a Gamma random variable can be approximated by a normal random variable with same mean and variance, if the shape parameter is large enough. Because in condition two, we already require τ to large enough to make the mean of K_b' large enough (see Eq. (3.32)). Hence, under the condition two, the random variable $\text{Gamma}(K_b', \beta)$ can also be approximated by the normal random variable with same mean and variance. According to Eq. (5.27) and Eq. (5.28), it means:

$$\text{Gamma}(K_b', \beta) \approx \text{Normal}(\beta a_b(\mathbf{x}_t) \tau, 2\beta^2 a_b(\mathbf{x}_t) \tau), \quad (5.29)$$

if τ is large enough to satisfy the condition two. Plugging Eq. (5.29) into Eq. (5.16), we obtain:

$$\begin{aligned} X_p(t + \tau) = X_p(t) + \sum_{j=1; j \neq b}^M \text{Normal}(a_j(\mathbf{x}_t) \tau, a_j(\mathbf{x}_t) \tau) v_{jp} \\ + \text{Normal}(\beta a_b(\mathbf{x}_t) \tau, 2\beta^2 a_b(\mathbf{x}_t) \tau). \end{aligned} \quad (5.30)$$

By linear combination theorem for normal random variables (see Eq. (3.35)), we can rewrite Eq. (5.30) into:

$$\begin{aligned} X_p(t + \tau) = X_p(t) + \left[\sum_{j=1; j \neq b}^M v_{jp} a_j(\mathbf{x}_t) \tau + \sum_{j=1; j \neq b}^M v_{jp} \sqrt{a_j(\mathbf{x}_t) \tau} N_j(0,1) \right] \\ + [\beta a_b(\mathbf{x}_t) \tau + \beta \sqrt{2a_b(\mathbf{x}_t) \tau} N_b(0,1)]. \end{aligned} \quad (5.31)$$

Simply substitute the notation τ with dt , and suppose that each dt has same size and satisfies both conditions, we finally arrive at the Langevin equation that describes burst, which has the form:

$$\begin{aligned}
 X_p(t+dt) = X_p(t) + [& \underbrace{\sum_{j=1; j \neq b}^M v_{jp} a_j(\mathbf{x}_t) dt}_{\text{increment from non-burst channel}} + \underbrace{\sum_{j=1; j \neq b}^M v_{jp} \sqrt{a_j(\mathbf{x}_t) dt} N_j(0,1)}_{\text{noise increment from non-burst channel}} \\
 & + [\underbrace{\beta a_b(\mathbf{x}_t) dt}_{\text{increment from burst channel}} + \underbrace{\beta \sqrt{2a_b(\mathbf{x}_t) dt} N_b(0,1)}_{\text{noise increment from burst channel}}].
 \end{aligned} \tag{5.32}$$

This equation describe the approximated stochastic increment for the reactant species S_p which is involved in a burst production channel. On the right side of Eq. (5.31), the terms inside first bracket describe the original Langevin approximation by Ref.[19], which is the same as that discussed in section 3.3. The terms inside second bracket describe the Langevin approximation for burst production channel R_b , with the exponentially distributed burst size which has mean β . We found that the difference between burst channel and non-burst channel is that, the burst channel has larger noise term $\beta \sqrt{2a_b(\mathbf{x}_t) dt} N_b(0,1)$ than a non-burst one $v_{jp} \sqrt{a_j(\mathbf{x}_t) dt} N_j(0,1)$. In other words, if β describes a deterministic uniform size of state change rather than an average of exponential distribution, it will simply go back to the non-burst Langevin form. The contribution of the factor $\sqrt{2}$ originates from Eq. (5.26), which takes $Var(K_b) = E[K_b]$ because of K_b is a Poisson random variable in our case. The factor $\sqrt{2}$ also characterize the higher noise contribution of a burst channel, compared to the other non-burst channels.

In short, the scheme of derivation can be summed up in the next figure:

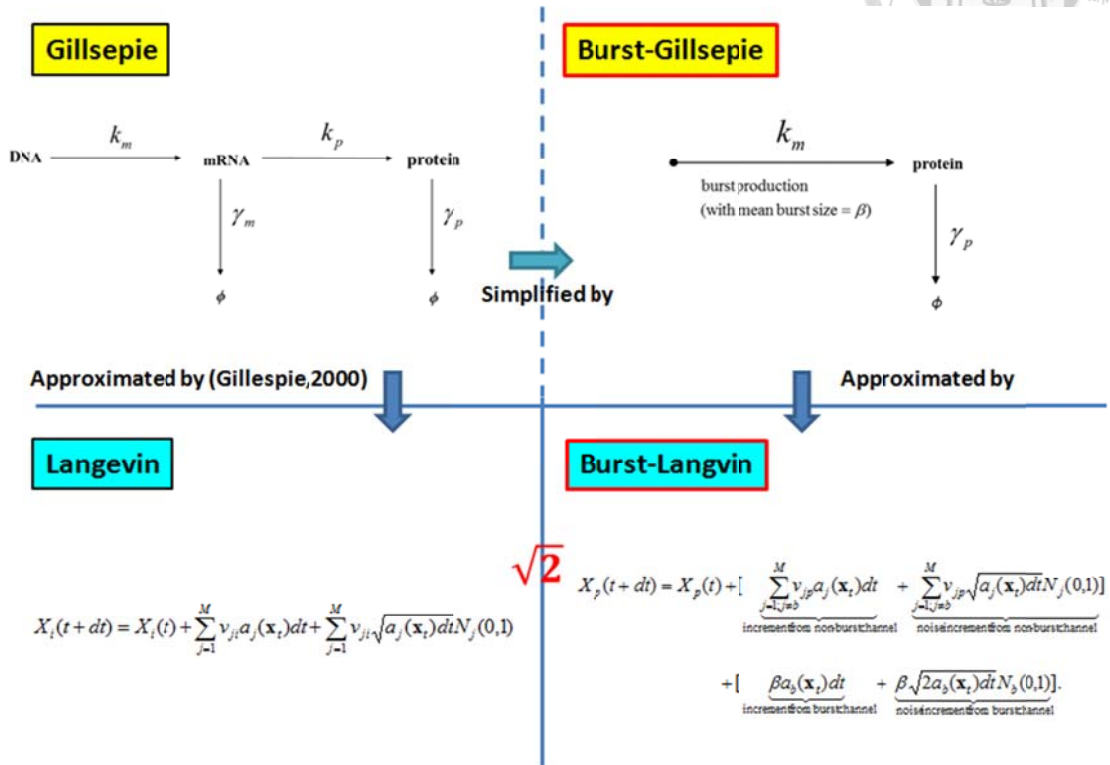


Figure 5-6 A schematic description for two-step model (left) or a bursting one-step model (right), and for Gillespie algorithm (upper panels) and their corresponding Langevin equation (lower panels).

To model the stochastic process of single gene expression, we first simplify the original two-step model (upper-left in Figure 5-6) into burst model (upper-right in Figure 5-6) to bypass the low copy number of mRNA molecules. Next, we derive the approximated Langevin form of burst model (lower-right in Figure 5-6, the burst-Langevin), and we find out the exponentially distributed burst size provide the $\sqrt{2}$ times of noise strength than the non-burst case (lower-left in Figure 5-6).

5.3 Numerical Validation of the Langevin form in burst production model



We notice that since the Langevin form in Eq. (5.32) “directly” describe the protein level in single gene expression model, and the description of mRNA level is effectively included in the mean burst size β . It means that Eq. (5.32) bypass the direct description of mRNA and avoid the problems in rare event case. Owing to this reason, Eq. (5.32) has a more robust simulating ability in rare event cases, and it successfully describes the original two-step model in single gene expression. The following figures show the simulation result of the single gene expression model using parameter set in Table 4-1.

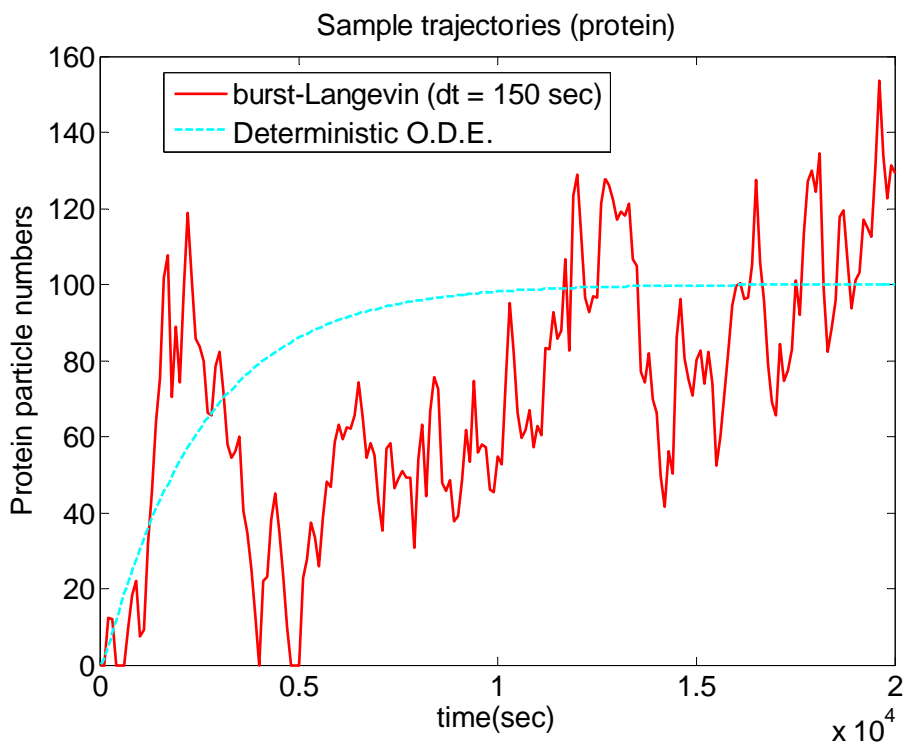


Figure 5-7 A sample trajectory for Langevin simulation in burst model. The parameters used in simulation are listed in Table 4-1 of section4.1. The time step used in simulation is 150 sec.

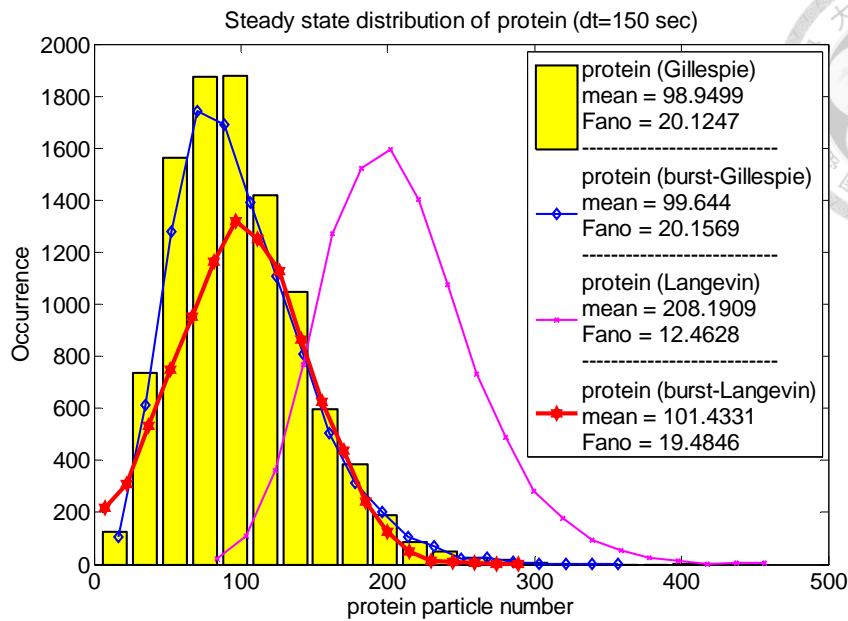


Figure 5-8 Steady state distribution of protein in different simulation approaches. Shown are distributions from 10,000 trajectories simulated with the parameters listed in Table 4-1 of section 4.1. The time step used in both Langevin forms is 150 sec.

The distributions in Figure 5-8 compare different simulating approaches and it shows a better approximation of Langevin in burst model, compared to the Langevin in two step which suffers from rare events problem in mRNA discussed in Chapter 4.

The scanning of the accuracy of approximation for different transcription rate and different dt is shown in the next figure (Figure 5-9). It shows different error levels of approximation under two different Langevin form. Left column: Langevin form in two-step model. Right column: Langevin form in burst model. The figure is generated by scanning the combination of (k_m, dt) and compare the result to analytical solution of the master equation. The rest of parameters are the same as Table 4-1. It shows that burst-Langevin not only provides a better approximation, but also stay more stable when transcription rate k_m is changing. It means that for simulating gene regulation, the burst-Langevin works better than original Langevin in two-step model.

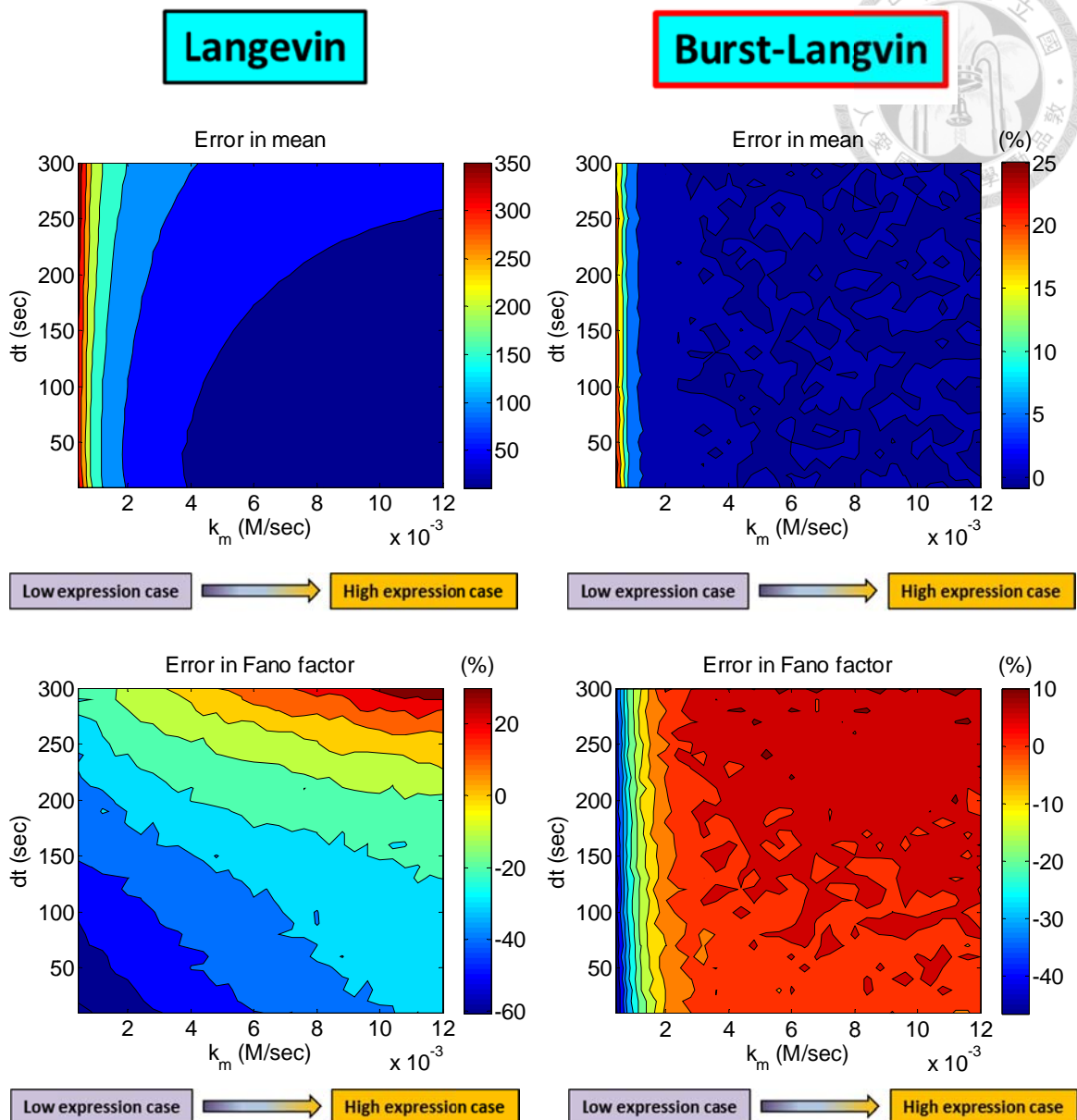


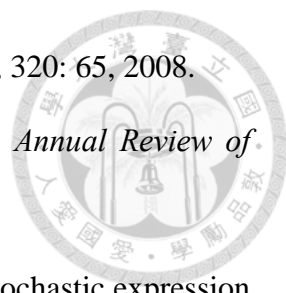
Figure 5-9 Comparison of different error levels in two different Langevin forms.

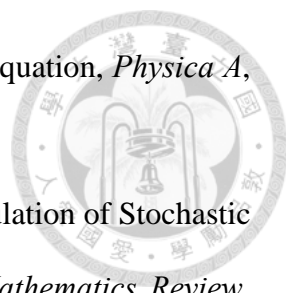
Shown are error in Mean (upper panels) and in Fano factor (lower panels) for traditional Langevin (Eq. (3.37) in section 3.3.2 or Eq. (4.9) in section 4.2, panels to the left) and the Burst-Langevin (Eq. (5.32) in section 5.2.2, panels to the right) in simulating a single gene expression model with the parameters listed in Table 4-1. The error are the difference between the statistic values of proteins from 10,000 trajectories as compared to analytical results at steady state (Eq. (4.8) in section 4.1).

REFERENCE



- [1] A. Raj and A. van Oudenaarden, Nature, Nurture, or Chance: Stochastic Gene Expression and Its Consequences, *Cell*, 135: 216-226, 2008.
- [2] M. B. Elowitz, A. J. Levine, E. D. Siggia and P. S. Swain, Stochastic Gene Expression in a Single Cell, *Science*, 297: 1183-1186, 2002.
- [3] A. Raj, S. A. Rifkin, E. Andersen, and A. van Oudenaarden, Variability in gene expression underlies incomplete penetrance, *Nature*, 463: 913–918, 2010
- [4] T. Kalmar, C. Lim, P. Hayward, S. M. Descalzo, J. Nichols, J. G. Ojalvo, A. M. Arias, Regulated Fluctuations in Nanog Expression Mediate Cell Fate Decisions in Embryonic Stem Cells, *Public Library of Science Biology*, 7(7): e1000149, 2009.
- [5] G. Chalancon, C. N. Ravarani, S. Balaji, A. Martinez-Arias, L. Aravind, R. Jothi and M. M. Babu, Interplay between gene expression noise and regulatory network, *Trends in Genetics*, 28(5): 221-232, 2012.
- [6] S. Hooshangi, S. Thiberge, and R. Weiss, Ultrasensitivity and noise propagation in a synthetic transcriptional cascade, *Proceedings of the National Academy of Sciences*, 102(10): 3581-3586, 2005.
- [7] M. Thattai and A. van Oudenaarden, Attenuation of Noise in Ultrasensitive Signaling Cascades, *Biophysical Journal*, 82: 2943-2950, 2002.
- [8] J. M. Pedraza and A. van Oudenaarden, Noise Propagation in Gene Networks, *Science*, 307: 1965-1969, 2005.
- [9] A. Eldar and M. B. Elowitz, Functional roles for noise in genetic circuits, *Nature*, 467:167–173, 2010.

- 
- [10] R. Losick and C. Desplan, Stochasticity and Cell Fate, *Science*, 320: 65, 2008.
- [11] C. J. Davidson and M. G. Surette, Individuality in Bacteria, *Annual Review of Genetics*, 42: 253–268, 2008.
- [12] Z. Hensel, H. Feng, B. Han, C. Hatem, J. Wang and J. Xiao, Stochastic expression dynamics of a transcription factor revealed by single-molecule noise analysis, *Nature Structural and Molecular Biology* 19: 797-802, 2012.
- [13] E. B. Jacob and D. Schultz, Bacteria determine fate by playing dice with controlled odds, *Proceedings of the National Academy of Sciences*, 107(30): 13197-13198, 2010.
- [14] U. Alon, Network motifs: theory and experimental approaches, *Nature Reviews Genetics*, 8: 450-461, 2007.
- [15] N. G. van Kampen, *Stochastic Processes in Physics and Chemistry 3rd edition*, Amsterdam, The Netherlands, 2007.
- [16] N. Maheshri and E. K. O'Shea, Living with Noisy Genes: How Cells Function Reliably with Inherent Variability in Gene Expression, *Annual Review of Biophysics and Biomolecular Structure*, 36: 413–434, 2007.
- [17] D. T. Gillespie, A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions, *Journal of Computational Physics*, 2:403-434, 1976.
- [18] D. T. Gillespie, Exact Stochastic Simulation of Coupled Chemical Reactions, *The Journal of Physical Chemistry*, 81(25): 2340, 1977.
- [19] D. T. Gillespie, The chemical Langevin equation, *Journal of Chemical Physics*, 113: 297, 2000.
- [20] D. A. McQuarrie, Stochastic Approach to Chemical Kinetics, *Journal of Applied Probability*, 4: 413-478, 1967.

- 
- [21] D. T. Gillespie, A rigorous derivation of the chemical master equation, *Physica A*, 188: 404-425, 1992.
- [22] D. J. Higham, An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations, *Society for Industrial and Applied Mathematics Review*, 43(3): 525–546, 2001.
- [23] D. T. Gillespie, Approximate accelerated stochastic simulation of chemically reacting systems, *Journal of Chemical Physics*, 115: 1716, 2001.
- [24] D. T. Gillespie, Perspective: Stochastic algorithms for chemical kinetics, *Journal of Chemical Physics*, 138: 170901, 2013.
- [25] M. Thattai and A. van Oudenaarden, Intrinsic noise in gene regulatory networks, *Proceedings of the National Academy of Sciences*, 98(15): 8614-8619, 2001.
- [26] N. Friedman, L. Cai, and X. S. Xie, Linking Stochastic Dynamics to Population Distribution: An Analytical Framework of Gene Expression, *Physical review letters (PRL)*, 97: 168302, 2006.
- [27] J. Paulsson, Summing up the noise in gene networks, *Nature* 427: 415-418, 2004.
- [28] E. M. Ozbudak, M. Thattai, I. Kurtser, A. D. Grossman and A. van Oudenaarden, Regulation of noise in the expression of a single gene, *Nature Genetics*, 31: 69 – 73, 2002.
- [29] I. Golding, J. Paulsson, S. M. Zawilski, and E. C. Cox, Real-Time Kinetics of Gene Activity in Individual Bacteria, *Cell*, 123(6): 1025-1236, 2005.
- [30] L. Cai, N. Friedman and X. S. Xie, Stochastic protein expression in individual cells at the single molecule level, *Nature*, 440: 358-362, 2006.
- [31] J. Yu, J. Xiao, X.J. Ren, K. Q. Lao, X. S. Xie, Probing Gene Expression in Live Cells, One Protein Molecule at a time, *Science* 311: 1600-1603, 2006.
- [32] A. B. Even, J. Paulsson, N. Maheshri, M. Carmi, E. O’Shea, Y. Pilpel and N.

Barkai, Noise in protein expression scales with natural protein abundance, *Nature Genetics*, 38: 636, 2006.

[33] J. R. Newman, S. Ghaemmaghami, J. Ihmels, D. K. Breslow, M. Noble, J. L. DeRisi and J. S. Weissman, Single-cell proteomic analysis of *S. cerevisiae* reveals the architecture of biological noise, *Nature*, 441: 840, 2006.

[34] J. M. Pedraza and J. Paulsson, Effects of Molecular Memory and Bursting on Fluctuations in Gene Expression, *Nature*, 455: 485–490, 2008.

[35] A. Maaref and R. Annavajjala, The Gamma Variate with Random Shape Parameter and Some Applications, *Institute of Electrical and Electronics Engineers Communications Letters*, 14(12): 1146-1148, 2010.