IMPACT OF STOCHASTIC TRANSCRIPTIONAL DELAY ON GENE NETWORKS.

A Dissertation

Presented to

the Faculty of the Department of Mathematics

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

José Manuel López Rodríguez

August 2014

IMPACT OF STOCHASTIC TRANSCRIPTIONAL DELAY ON GENE NETWORKS.

José Manuel López Rodríguez

APPROVED:

Dr. Krešimir Josić,
Chairman
Department of Mathematics, University of Houston

Dr. William Ott
Department of Mathematics, University of Houston

Dr. Zachary P. Kilpatrick
Department of Mathematics, University of Houston

Dr. Matthew R. Bennett
Department of Biochemistry and Cell Biology,
Rice University
Institute of Biosciences and Bioengineering,
Rice University

Dean, College of Natural Sciences and Mathematics

Acknowledgements

First I would like to thank my advisor, Dr. Krešimir Josić, for believing that I would make it through this journey and never giving up on me. If it were not for his patience, advice and guidance I would not have acquired the knowledge and skills that I have today. I cannot thank him enough for all the help he has provided me ever since I started learning from him as an undergraduate. I can only hope that some of his many great qualities may have rubbed off on me.

I would like to thank my academic collaborators Dr. William Ott, Dr. Chinmaya Gupta, Dr. Matthew Bennett and Dr. LieJune Shiau. I especially extend my appreciation to Dr. Gupta and Dr. Ott for helpful discussions during the creation of much of the software I created.

I would also like to thank my professors and peers within the Department of Mathematics at the University of Houston. Special thanks to my friends James West, Satish Pandey, Burcin Ozcan and Bilge Kayasandik for their valuable help in understanding the concepts covered in our classes.

Words cannot express my gratitude to my parents and siblings for understanding how difficult and time-consuming graduate school has been on me. I would like to apologize from the bottom of my heart to my mother for not being able to spend as much time with her as I did while growing up. To my father, I want to thank him for always being there for me and for making me the man I am today.

To my close friend James Trousdale, thank you for the support and friendship over the years and for the high expectations that have pushed me to achieve more than I thought I

could.

Lastly, I want to thank my fiancée Daisy Cherian. Thank you for sticking by my side and motivating me to keep going. Your love and encouragement gave me the endurance to see this day become a reality.

IMPACT OF STOCHASTIC TRANSCRIPTIONAL DELAY ON GENE NETWORKS.

An Abstract of a Dissertation

Presented to

the Faculty of the Department of Mathematics

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

José Manuel López Rodríguez

August 2014

Abstract

The creation of protein from DNA is a dynamic process consisting of numerous components that include transcription, translation and protein folding. Each of these components is further comprised of hundreds or thousands of sub-steps that must be completed before a fully mature protein is formed. Consequently, the time it takes to create a single protein depends on the number of steps in the reaction chain and the nature of each step. Instead of modeling each of these steps in detail, one way to account for these reactions in models of gene regulatory networks is to incorporate dynamical delay. The stochastic nature of the reactions necessary to produce protein leads to a waiting time that is randomly distributed, complicating simulation and analysis.

We examine this problem using different examples and approaches. First, we describe how queueing theory can be used to examine the effects of such distributed delay on the propagation of information through transcriptionally regulated genetic networks. In an analytically tractable model we find that increasing the variance in protein production delay while holding the mean fixed increases signaling speed in transcriptional networks. The effect is confirmed in stochastic simulations, and we demonstrate its impact in several common transcriptional motifs. Next we examine how such delay affects bistable systems. We investigate several stochastic models of bistable gene networks and find that increasing delay dramatically increases the mean residence times near stable states. We show that this behavior can be explained using a non-Markovian, analytically tractable reduced model. Finally, we explore the relationship between delay birth-death processes and their appropriate approximating delay chemical Langevin equations. Simulations demonstrate that, if done correctly, a delay chemical Langevin approximation is accurate even at moderate system sizes.

Together, these results provide a foundation for the implementation of detailed stochastic simulation algorithms in the study of the delay stochastic processes that model biochemical networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A gene network is a collection of DNA sections that interact with each other through the proteins they express. Patterns of gene network activity are fundamental to life. Understanding gene network behavior can therefore provide us with valuable insights about the dynamical processes within living beings.

Gene regulation forms a basis for cellular decision-making processes, and transcriptional signaling is one way in which cells can modulate gene expression patterns [44]. The intricate networks of transcription factors and their targets are of intense interest to theorists because it is hoped that topological similarities between networks will reveal functional parallels [1]. Models of gene regulatory networks have taken many forms, ranging from simplified Boolean networks [20, 46], to full-scale stochastic descriptions simulated using Gillespie's algorithm [28].

Gene networks are frequently modeled using systems of nonlinear ordinary differential equations. While such models offer insights about certain aspects of gene network dynamics, they fail to capture the stochasticity inherent to the dynamical processes that

occur within living cells. Many ignored reactions, like oligomerization of transcription factors or enzyme-substrate binding, occur at much faster timescales than reactions such as transcription and degradation of proteins. Reduced models are frequently obtained by eliminating these fast reactions [9, 13, 47, 51]. Such reductions can still introduce uncontrolled errors, even if done correctly. For instance, a linear sequence of reactions that are not explicitly modeled can introduce an effective delay in the network. This type of behavior, often termed linear chain delay, has long been known to exist in gene regulatory networks. However, delay is frequently not included in models of gene networks, and its effects on gene network dynamics is not fully understood.

In protein production, delay could occur as the result of the sequential assembly of first mRNA and then the protein (Fig. 1.1) [34, 56, 64]. This process begins when the enzyme RNA polymerase binds to a gene and begins its transcription. This gives rise to a sequence of nucleotides called mRNA. Some mRNAs may need to be transported from the nucleus to the cytoplasm where a ribosome may bind to it in order to begin its translation. This results in an amino acid which then undergoes several reactions with other amino acids in order to create the protein. By assuming that a single mRNA will give rise to a protein in a random amount of time and ignoring all the reactions that take place in the production, we can quantify delay. Preliminary results point to the delay being between 5 and 20 minutes. These numbers can depend on many biological factors.

Delayed differential equations (DDEs) have been used to address this problem as an alternative to ordinary differential equations (ODE) models. Delay can qualitatively alter the local stability of genetic regulatory network models [19] as well as their dynamics, especially in those containing feedback. For instance, in models of transcriptional negative feedback, delay can lead to oscillations in the protein expression [3, 34, 55, 61, 65, 78], and

experimental evidence suggest that oscillations in simple synthetic networks are due to transcriptional delay [81,86]. These oscillations may result due to the burstiness-like effect due to delay. The system may be at rest while protein production takes place. This results in a large number of proteins binding to the gene that produces them, effectively repressing its own production. Eventually degradation will eliminate these proteins allowing for production once again, thus repeating the cycle. Further, it has been shown that delayed repression is required for circadian oscillations in mammalian cells [90]. Other studies have investigated the relationship between delay and stochasticity [10,35,76].

Protein production delay times are difficult to measure in live cells, though recent work has shown that the time it takes for transcription to occur in yeast can be on the order of minutes and is highly variable [52]. Still, transcriptional delay is thought to be important in a host of naturally occurring gene networks. For instance, mathematical models suggest that circadian oscillations are governed by delayed negative feedback systems [79, 80], and this was experimentally shown to be true in mammalian cells [90]. Delay appears to play a role in cell cycle control [18, 77], apoptosis induction by the p53 network [85], and the response of the NF-κB network [65]. Delay can also affect the stochastic nature of gene expression, and the relation between the two can be subtle and complex [10,35,57,76]. Delay can also play an important role in determining the stability of biological switches.

Bistability is a central characteristic of biological switches. It is essential in the determination of cell fate in multicellular organisms [42], the regulation of cell-cycle oscillations during mitosis [40], and the maintenance of epigenetic traits in microbes [67]. Due to the stochastic nature of gene expression, bistable gene networks can randomly switch between stable states [47]. This phenomenon has been studied in many contexts, including

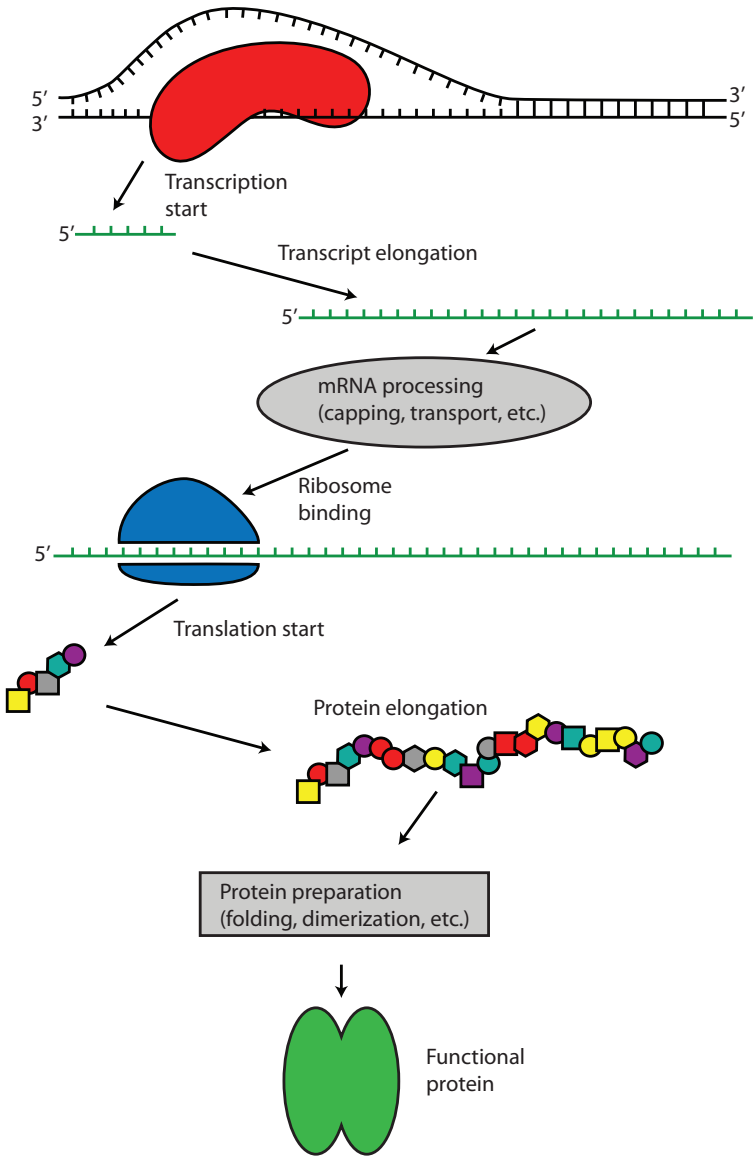

Figure 1.1: **Delay source in protein production**. In order for a mature protein to be produced, the gene must first be transcribed and the mRNA must then be translated. Each of these processes is comprised of hundreds or thousands of reactions that must occur as the polymerase or ribosome progress through the sequence of nucleotides. Figure by Matthew R. Bennett.

the lysis/lysogeny switch of bacteriophage λ [5,95], bacterial persistence [6], and synthetically constructed positive feedback loops [26,66].

While delay and bistability have been extensively studied, the impact of transcriptional delay on bistable gene networks is not clear. In general, the interaction between delay and stochasticity is complex. Modeling suggests delay affects the stochastic properties of gene expression [10, 35, 57, 76], and that stochastic delay can accelerate signaling in genetic pathways (as we will see in Chapter 3). The effect of delay on the mean first-passage times for Langevin-approximations to bistable gene networks has been studied, but no general principles have been proposed. Many results appear to contradict each other [45, 53, 74, 93, 94]. There are many possible reasons for these discrepancies. For instance, Langevin approximations can fail at small system sizes typically found inside cells, and *ad hoc* choices of Langevin models can lead to different predictions for the same underlying system.

Given the importance of delay in gene regulatory networks, it is necessary to develop methods to simulate and analyze such systems across spatial scales. In the absence of delay, it is well known that chemical reaction networks are accurately modeled by ordinary differential equations (ODEs) in the thermodynamic limit, *i.e.* when molecule numbers are sufficiently large. When molecule numbers are small, however, stochastic effects can dominate. In this case, the chemical master equation (CME) describes the evolution of the probability density function over all states of the system to which Gillespie's stochastic simulation algorithm (SSA) samples trajectories from the probability distribution described by the CME.

While exact, the CME is difficult to analyze and the SSA can be computationally expensive. To address these issues a hierarchy of coarse-grained approximations of the SSA has been developed [41] (see Figure 1.2). Spatially discrete approximations such as τ-leaping [16, 17, 30, 96] and K-leaping [14] trade exactness for efficiency. At the next level

are chemical Langevin equations (CLEs), which are stochastic differential equations of dimension equal to the number of species in the biochemical system. CLEs offer two advantages. First, unlike the SSA, the well-developed ideas from random dynamical systems and stochastic differential equations apply to CLEs. Second, it is straightforward to simulate large systems using CLEs. Finally, in the thermodynamic limit, one arrives at the end of the Markovian hierarchy: the reaction rate equation (RRE).

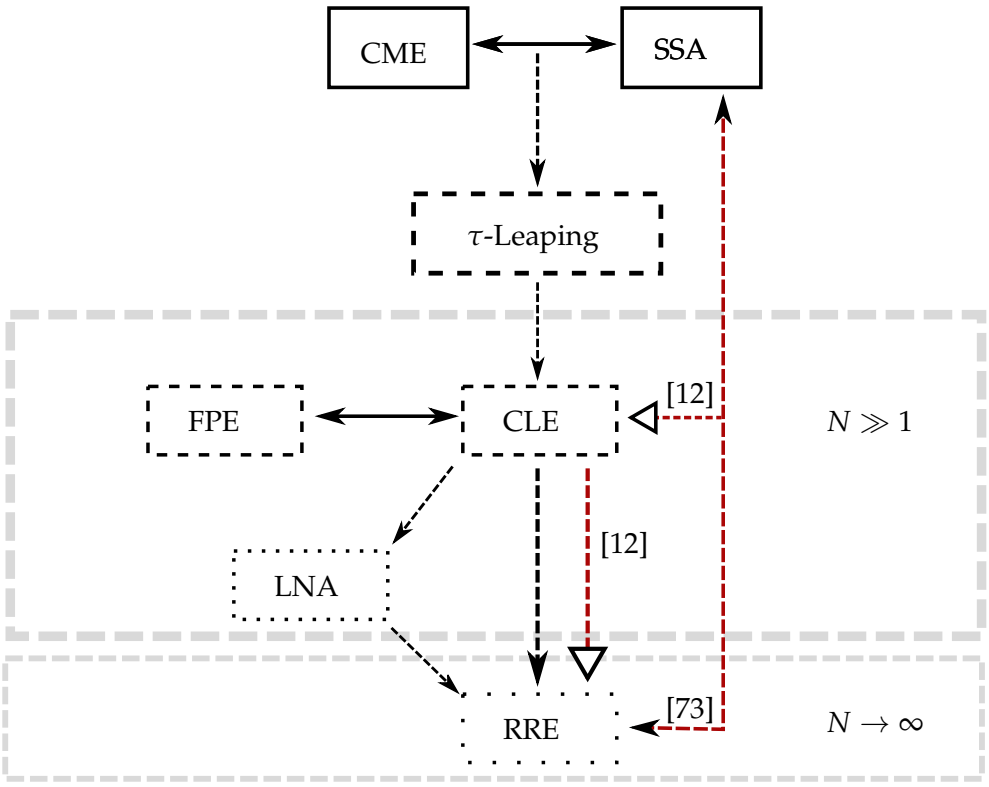

Figure 1.2: **Schematic of the modeling hierarchy for biochemical systems**. The black arrows link the various components of the theory for Markov systems (no delay); red arrows link the corresponding delay components. Numbers attached to arrows refer to papers that establish the corresponding links. Empty arrowheads denote heuristic derivations; in this paper, we rigorously establish the dSSA to dCLE and dCLE to dRRE links.

The Markovian hierarchy mentioned above (no delay) is well-understood [31,41], but a complete analogue of the Markovian theory does not yet exist for systems with delay. The SSA was generalized to a delay version - the dSSA - to allow for both fixed and variable delay as we will see in Chapter 2. Notably, the modification of the SSA to incorporate non-Markov processes pre-dates the dSSA [27]; here many exponential steps from a linear chain are replaced by a gamma-distributed delay. Some analogues of τ-leaping exist for systems with delay; see *e.g.* D-leaping [7].

Several methods have been used to formally derive a delay chemical Langevin equation (dCLE) from the delay chemical master equation (dCME). Brett and Galla [12] use the path integral formalism of Martin, Siggia, Rose, Janssen, and de Dominicis to derive a dCLE approximation without relying on a master equation. The Brett and Galla derivation produces the 'correct' dCLE approximation of the underlying delay birth-death (dBD) process in the sense that the first and second moments of the dCLE match those of the dBD process. However, their derivation has some limitations. In particular, it gives no rigorous quantitative information about the distance between the dBD process and the dCLE.

In Chapter 2 we present a description of the stochastic simulation algorithm that is applied to the models used throughout the work. In Chapter 3 we explore the effects of randomly distributed delay on simple gene regulatory networks. We confirm several analytical results by using an exact stochastic simulation algorithm that takes variability within the delay time into account - the delayed stochastic simulation algorithm or dSSA. One of the main findings is that when the mean of the delay time is fixed, the mean time required for downstream signaling decreases as we increase delay variability.

In Chapter 4 we investigate several bistable gene networks using the dSSA: a single-gene positive feedback loop; the co-repressive toggle switch [26]; and the lysis/lysogeny

switch of phage λ [95]. We find that although the details and dimensionality of the networks differ, in each the mean residence times near the stable states *increase* dramatically with even modest increases in transcriptional delay time.

Finally, in Chapter 5 we present the correct dCLE approximation and observe that it performs remarkably well at moderate system sizes in a number of dynamical settings: steady state dynamics, oscillatory dynamics, and certain metastable switches.

The reader may also find it helpful to look over the Appendix which contains a description and solutions of several concrete problems that appeared during the completion of several projects discussed in this work. In particular, it contains a description of how to deal with data transfering among several collaborators as well as a description of a document syntax which enables access to dynamic libraries written in `C++` from scripting languages such as MATLAB and Python.

Chapter 2

Stochastic Simulations

Many real-world systems can change at any point in time, but experience discrete changes in state. Examples include groups of distinct animal as well as molecular species that interact with one another. Here, we address examples where changes in population size are random. Such changes are consequences an occurrence happening at a determinable time and place, an "event". An event could be a birth that increases the number of animals of a given species by one. Similarly, a binding of a substrate to an enzyme would reduce the number of both molecules by one, increasing the number of molecules of the resulting compound by one. Thus, an event if a general term for anything that changes the state of a system.

Deterministic models have been developed in many sciences to answer two main questions about such systems: Given that we start with a system of N distinct species:

1. What will be the population levels in each species after a given amount of time?

2. How long will it take for a population to reach a given state?

Deterministic systems can approximate the behavior of a stochastic system in certain limits. In the limit of a large population a deterministic model can provide a good description of the evolution of the density of molecules in a well mixed system. However, these approximations typically only hold over a finite time, and can break down when the number of molecules is small.

To be able to track the state of a system accurately, we must therefore account for the randomness involved. This means that we can either simulate different realizations of a system using Monte Carlo models, or describe how the probability densities for the different populations evolve in size. This is known as stochastic modeling. Note that while for a deterministic model every simulation gives the same result, Monte Carlo simulations produce a different realization every time. We therefore use a statistical description of the system.

There are different ways to simulate stochastic processes of this type. However, most of these methods share a basic structure. In the following sections we will discuss this shared structure and the stochastic methods that we will use in this work.

2.1 Simulation Methods

In a population process we keep track of the state of the system and the time. Since a Monte Carlo simulation of a population evolving in time is based on iteration, we will let the subscript in a variable denote the iteration number for the remainder of this chapter. This allows us to refer to each of the points generated during a simulation. For instance, we let (t_i, X_i) denote the time and the state of the system after the i-th iteration and τ_i the time spent in the interval between iterations, i.e. $\tau_i = t_i - t_{i-1}$. If X_i denotes the state

vector then $X_i[j]$ denotes the size of the j-th population during the i-th iteration.

We note that the stochastic processes we will be describing do not always satisfy the Markov property. This means that in certain cases the future of the system will not depend only on the present state, but may also depend on the history.

We should observe that the data obtained from the simulations will need to be processed to be interpreted. Suppose that a stochastic simulation has been run for m iterations. Given a number ϕ less than t_m, how can we estimate the state of the system at time ϕ if for each positive integer $i < m$, $t_i \neq \phi$? The answer depends on the simulation method used. A simple approximation that can be used regardless of the method is a linear interpolation. We must keep in mind that if our simulation deals with discrete states then we must round our answers to the nearest integer. To obtain the state we need to find an integer n that satisfies $t_{n-1} < \phi < t_n$. Once such integer has been found then the state of the system at time ϕ is given by

$$\begin{cases} \lfloor E \rfloor & \text{if } X_n > X_{n-1} \\ \lceil E \rceil & \text{if } X_n < X_{n-1} \end{cases} \qquad \text{where } E = \frac{(X_n - X_{n-1})}{\tau_n}(\phi - t_n) + X_n \tag{2.1}$$

In the previous equations $\lfloor a \rfloor$ denotes the largest integer less than or equal to a and $\lceil a \rceil$ denotes the smallest integer greater than or equal to a. These are commonly known as the floor and ceiling functions, and can be used if the system under study has a discrete state.

The right panel of 2.1 demonstrates the results of such a discrete interpolation. As we will see in the next section, this plot displays an accurate path describing the sizes of a given population.

It was mentioned at the beginning of the chapter that all simulations share a basic structure. This structure consists of the sets $X = \{X_1, ..., X_i\}$, $t = \{t_1, ..., t_i\}$ and $\tau =$

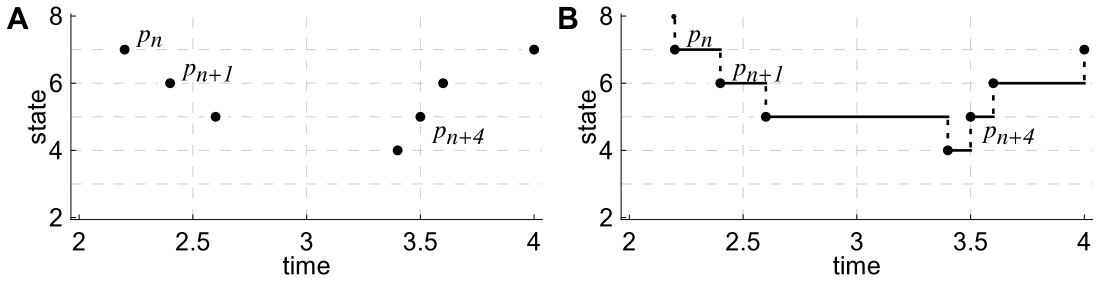

Figure 2.1: **Example of "discrete interpolation" between points**. **(A)** Unprocessed raw data obtained from a simulation shows a few points for the trajectory of a given population. Here $p_n = (t_n, X_n)$ denotes the state of the system during the n-th iteration. **(B)** To estimate the state of the system at any given time we can apply (2.1) to each pair of consecutive points.

$\{\tau_1, ..., \tau_i\}$ together with two functions to determine how the system evolves. That is, we need a function to determine the change in time and a function to determine the change in the state. With this in mind we can now proceed to explore the next two algorithms.

As mentioned at the beginning of the chapter many stochastic systems can be described by the following set of objects: The sets $X = \{X_1, ..., X_i\}$, $t = \{t_1, ..., t_i\}$ and $\tau = \{\tau_1, ..., \tau_i\}$ together with two functions to determine how the system evolves. That is, we need a function to determine the change in time and a function to determine the change in the state. With this in mind we can now proceed to explore two specific simulation algorithms.

2.2 Stochastic Simulation Algorithm (SSA)

In 1977 Daniel T. Gillespie introduced the exact Stochastic Simulation Algorithm (SSA). By "exact" we mean that this method provides sample trajectories from the exact family of probability distribution that define the stochastic process. The algorithm is based on

two main computational steps: (1) Given that the occurrence of an event was observed at time t_n, obtaining the waiting time τ_{n+1} to the next event, and (2) Determining what next event takes place at time t_{n+1}?

To complete the first step we must think in probabilistic terms. We must keep in mind that the interevent time – the time between two consecutive events – is random. It can be easily shown that the inter-event times are exponentially distributed

$$P(\tau_n; s_n) = \begin{cases} s_n \cdot e^{-s_n \tau_n} & \text{if } \tau_n \geq 0 \\ 0 & \text{if } \tau_n < 0 \end{cases} \tag{2.2}$$

where

$$s_n = \sum_{i=1}^{M} a[i](X_n) \tag{2.3}$$

Gillespie used the term a_0 instead of s_n to denote the sum of the propensity functions evaluated at the current state of the system. Recall that here we use a subscript to denote the iteration number and thus we are avoiding the use of subscripts for purposes other than the ones already stated.

The density function (2.2) tells us how to sample the interevent times exactly. It remains to determine which event takes place at the end of that time. Equation (2.3) is actually a consequence of a more general result. The probability that the μ-th event is seen once during an interval of time τ_n is

$$P(\tau_n, \mu; s_n) = \begin{cases} a[\mu] \cdot e^{-s_n \tau_n} & \text{if } \tau_n \geq 0 \\ 0 & \text{if } \tau_n < 0 \end{cases} \tag{2.4}$$

Equation (2.2) can be found from this result which implies

$$P(\mu; s_n) = \begin{cases} a[\mu]/s_n & \text{if } \mu \in \{1, 2, \cdots, M\}, \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

since $P(\tau_n, \mu; s_n) = P(\tau_n; s_n) \cdot P(\mu; s_n)$.

These probability density functions are very important to simulate the process. In every iteration of the simulation we need to obtain values τ and μ so that if we have a set of the times it takes for the system to switch from state X_n to X_{n+1}, then the set will follow the distribution in (2.2). To generate numbers that follow a particular probability distribution we can use a simple solution provided that the machine used to perform the simulation has access to a unit-interval uniform random number generator.

2.2.1 Pseudorandom Numbers

Digital computers do not produce truly random numbers. After all, they are designed to be precise and to follow sets of instructions. Nonetheless, there are many algorithms that can produce sequences of pseudorandom numbers which approximates the properties of random numbers [48]. Each sequence can be started with what is called a seed. The pseudorandom number generator will produce the same sequence of numbers if the seed is the same.

Given that we can sample a number from a uniform distribution, we can sample numbers from an arbitrary univariate distribution. In some cases however, it my be best to come up with a different procedure to generate other types of distributions. The idea is that if we can generate a set of numbers from the unit-interval such that the probability that a number lies in a subinterval $[a, b]$ is $1/(b-a)$, then we can use those numbers to generate any other type of distributions by first computing their cumulative distributions. This is because the cumulative distribution is a function that returns values from the unit-interval. Thus, if F is the cumulative distribution function of a random variable with probability density function f and r is a number in $[0, 1]$, then we can find a number

R such that

$$r = F(x) = \int_{-\infty}^{R} f(t)\, dt$$

in the case f is a continuous function, or

$$r \leq F(x) = \sum_{i=1}^{R} f(y_i)$$

in the discrete case. Note that in the discrete case the less than or equal sign allows for more than one integer R satisfying this condition. For this case we can choose the smallest integer R that meets this condition.

Finding an expression for R may not be an easy task. Nonetheless, we can easily find an expression for R if the probability density function is (2.2). Solving for R in

$$r = \int_{-\infty}^{R} P(t; s_n)\, dt = \int_{0}^{R} s_n \cdot e^{-s_n t} dt$$

gives

$$R = -\frac{1}{s_n} \ln(1 - r) \cong \frac{1}{s_n} \ln\left(\frac{1}{r}\right)$$

Technically, we should be using $1 - r$ instead of r in the last expression. This does not affect any of the results since the two expressions are uniformly distributed. That is, seeing that r is a random number in the interval $[0, 1]$, then so is the number $1 - r$.

2.2.2 Algorithm

We can now proceed to present Gillespie's method. Recall the main question on the development of the method: Given that the occurrence of an event was observed at time t_n, what is the waiting time τ_{n+1} on the state X_n for the next event to occur? What event takes place at time t_{n+1}? The waiting time τ_{n+1} is a random number from a distribution with probability density function (2.2) and the event that takes place is the event number

μ from the distribution with probability mass function (2.5). This leads us to the following algorithm:

Given an initial state for the system X_0 at time $t_0 = 0$ then the state X_n of the system and time t_n are defined to be

$$t_{n+1} = t_n + \tau_{n+1}, \qquad \tau_{n+1} = \frac{1}{s_n} \ln\left(\frac{1}{r_{n+1}}\right) \tag{2.6a}$$

$$X_{n+1} = X_n + z[\mu_{n+1}], \qquad \mu_{n+1} \text{ is the smallest integer satisfying } r'_{n+1} \leq \sum_{i=1}^{\mu_{n+1}} \frac{a[i]}{s_n} \tag{2.6b}$$

where r_n and r'_n denote random numbers from the unit-interval uniform distributions during the n-th iteration and s_n is the expression defined in (2.3).

Finally, we note that when we implement this algorithm, we must define when we want the procedure to stop executing. It can be forced to stop after a certain time limit w, that is, after the time of the m-th iteration satisfies $t_m > w$, or perhaps after one of the population in the system has reached a pre-determined level. Most importantly however, the execution has to stop if s_n is zero. When the sum of the propensity function is zero it means that none of the possible events can occur in the future since their probabilities are zero.

2.3 Delayed Stochastic Simulation Algorithm (DSSA)

The idea behind extending Gillespie's SSA is that if a reaction is to be delayed by some amount of time then we temporarily store this reaction along with the time at which the event will occur in a queue. At the time the reaction is scheduled to happen we pull it from the queue. See Algorithm 1 of [15] for a more efficient version of the algorithm we describe here. It should be noted that Algorithm 2 in [15] is the same as the one

described here but he omits describing the data structures used in the method in detail. Here we provide the definition of the data structure while providing a few insights about possible implementations. We should note that this algorithm is usually rediscovered using a heuristic modification on the SSA. To see an explanation on why this algorithm produces an exact simulation we refer you to Section III B in [15].

To implement this extension we can create two structures: a `delayedEvent` and an array of `delayedEvents` which we will call `DEArray`.

A `delayedEvent` is an ordered number pair. The first number denotes the time at which the delayed reaction will take place and the second one is the reaction number. A `DEArray` is an array of `delayedEvents`. We have to perform three operations on this array: `insert`, `pull` and `check`

- The `insert` operation must be able to insert a `delayedEvent` into the `DEArray`. Special care must be taken since the `DEArray` must be able to locate the `delayedEvent` with the smallest time. In other words, when inserting a `delayedEvent` into the array we must insert it into the `DEArray` in such a way that the `DEArray` is always in an increasing order according to the times of the `delayedEvents`.
- The `pull` operation will remove the `delayedEvent` with the smallest time and will return such `delayedEvent`.
- The `check` operation should return the time of the `delayedEvent` with the smallest time and in the case that the `DEArray` is empty it should return a number that represents infinity.

There are a few issues that must be considered when implementing a `DEArray`, speed and memory usage. Depending on the programming language that we are using we

might be able to create linked lists and/or arrays. Arrays can be declared to contain more elements than the number of data items expected but this can waste memory. Linked list can provide with better memory utilization. The insertion and deletion in the sorted array can be time consuming since all the elements following the inserted or deleted element must be shifted appropriately. Since the elements of array are stored contiguously in memory we are allowed to have access to any array element. In a linked list we do not afford such access. Using linked lists in the creation of a `DEArray` can save memory but we must keep in mind that dynamic allocation incurs the overhead of function calls and this can have some impact in the speed of our simulation. Having considered these issues we must keep in mind the the Delayed SSA can work as long as we have the three operations mentioned above.

The last element that we need is the delayed vector `dv`. This vector is of equal size as the vector of propensity functions, and must be evaluated during each iteration of the simulation. It is composed of *positive* random numbers from a given probability distribution. If the i-th event is not to be delayed then `dv[i]` must be set to zero.

2.3.1 Algorithm

Given the initial state for the system X_0 at time $t_0 = 0$ we proceed by creating an array of positive real numbers `dv` of length equal to the amount of events in the system as well as the `DEArray timeStack`.

Next we compute

$$\begin{aligned} \texttt{dv}_{n+1} &= \text{array of random numbers from delay distribution} \\ \tau_{n+1} &= \frac{1}{s_n} \ln\left(\frac{1}{r_{n+1}}\right) \end{aligned}$$

and call the `check` operation on `timeStack` which will let us know if there is an event scheduled to take place. If the $t_n + \tau_{n+1}$ is less than the time of the possible scheduled event (there may be no event scheduled) then we may proceed to compute the following expressions:

$$\begin{aligned} t_{n+1} &= t_n + \tau_{n+1} \\ r'_{n+1} &= \text{ random number from uniform unit distribution} \\ \mu_{n+1} &= \text{ smallest integer satisfying } r'_{n+1} \leq \sum_{i=1}^{\mu_{n+1}} \frac{a[i]}{s_n} \end{aligned}$$

If the μ_{n+1} event is to be delayed, that is if $\mathtt{dv}_{n+1}[\mu] > 0$ then we use the insert operation on `timeStack` to store the `delayedEvent` composed of the time $t_{n+1} + \mathtt{dv}_{n+1}[\mu_{n+1}]$ and the μ_{n+1} event. Otherwise we let

$$X_{n+1} = X_n + z[\mu_{n+1}]$$

In the case when $t_n + \tau_{n+1}$ is not less than the time of the scheduled event then we first need to check if `timeStack` is empty. This step is important and it is the equivalent of checking to see if s_n is not zero as in the SSA. When this is the case, we must stop the simulation for the same reason as mentioned in the previous in the SSA algorithm.

Finally, if `timeStack` is not empty then we pull the delayed event. These values are temporary, so assume that t_d is the time for the delayed event and μ_d is the event number that was delayed. Then we compute

$$\begin{aligned} \tau_{n+1} &= t_d - t_n \\ t_{n+1} &= t_d \\ X_{n+1} &= X_n + z[\mu_d]. \end{aligned}$$

The computation of τ_{n+1} in this last step is important since τ_{n+1} should tell us the time between two consecutive events. If we were to try to obtain data from this variable during

your simulation without taking care of this step we would obtain erroneous data (as we did during the implemention of the algorithm). Notice, that in this step we have overwritten the initial value of τ_{n+1}, for this reason this version of the delayed SSA is also called the Rejection Method.

Chapter 3

Stochastic Delay in Gene Networks

In this Chapter[1], we examine the consequences of randomly distributed delay on simple gene regulatory networks: We assume that the delay time for protein production, T, is not constant but instead a random variable. If f_T denotes the probability density function (PDF) of T, this situation can be described deterministically by an integro-delay differential equation [72] of the form

$$\dot{\mathbf{x}}(t) = \int \mathbf{g}[\mathbf{x}(t), \mathbf{x}(t-s)] f_T(s)\, \mathrm{d}s, \tag{3.1}$$

where $\mathbf{x}$ is a positive definite state vector of protein concentrations, and $\mathbf{g}$ is a vector function representing the production and degradation rates of the proteins. Note that processes that do not require protein synthesis (like dilution and degradation) will depend on the instantaneous, rather than the delayed, state of the system. Therefore $\mathbf{g}$ is in general a function of both the present and past state of the system.

[1] Content in this Chapter has been previously published: Josić, K., López, J. M., Ott, W., Shiau, L., Bennett, M. R. (2011). **Stochastic delay accelerates signaling in gene networks**. *PLoS computational biology*, 7(11), e1002264.

Equation (3.1) only holds in the limit of large protein numbers [72]. As protein numbers approach zero, the stochasticity associated with chemical interactions becomes non-negligible. Here, we address this issue by expanding on Eq. (3.1) using an exact stochastic algorithm that takes into account variability within the delay time [72]. We further use a queueing theory approach to examine how this variability affects timing in signaling cascades. We find that when the mean of the delay time is fixed, increased delay variability accelerates downstream signaling. Noise can thus increase signaling speed in gene networks. In addition, we find that in simple transcriptional networks containing feed-forward or feedback loops the variability in the delay time nontrivially affects network dynamics.

Queueing theory has recently been used to understand the behavior of genetic networks [4, 54, 62]. Here we are mainly interested in dynamical phenomena to which the theory of queues in equilibrium used in previous studies cannot be applied. As we explain in the next sections, gene networks can be modeled as thresholded queueing systems: Proteins exiting one queue do not enter another queue, as would be the case in typical queueing networks. Rather, they modulate the *rate* at which transcription is initiated, and thus affect the *rate* at which proteins enter other queues.

3.1 Distributed delay in protein production

The transcription of genetic material into mRNA and its subsequent translation into protein involves potentially hundreds or thousands of biochemical reactions. Hence, detailed models of these processes are prohibitively complex. When simulating genetic circuits it is frequently assumed that gene expression instantaneously results in fully formed proteins. However, each step in the chain of reactions leading from transcription initiation

to a folded protein takes time (Figure 3.1). Models that do not incorporate the resulting delay may not accurately capture the dynamical behavior of genetic circuits [61]. While earlier models have included either fixed or distributed delay [72,97,98], here we examine specifically the *effects* of delay variability on transcriptional signaling.

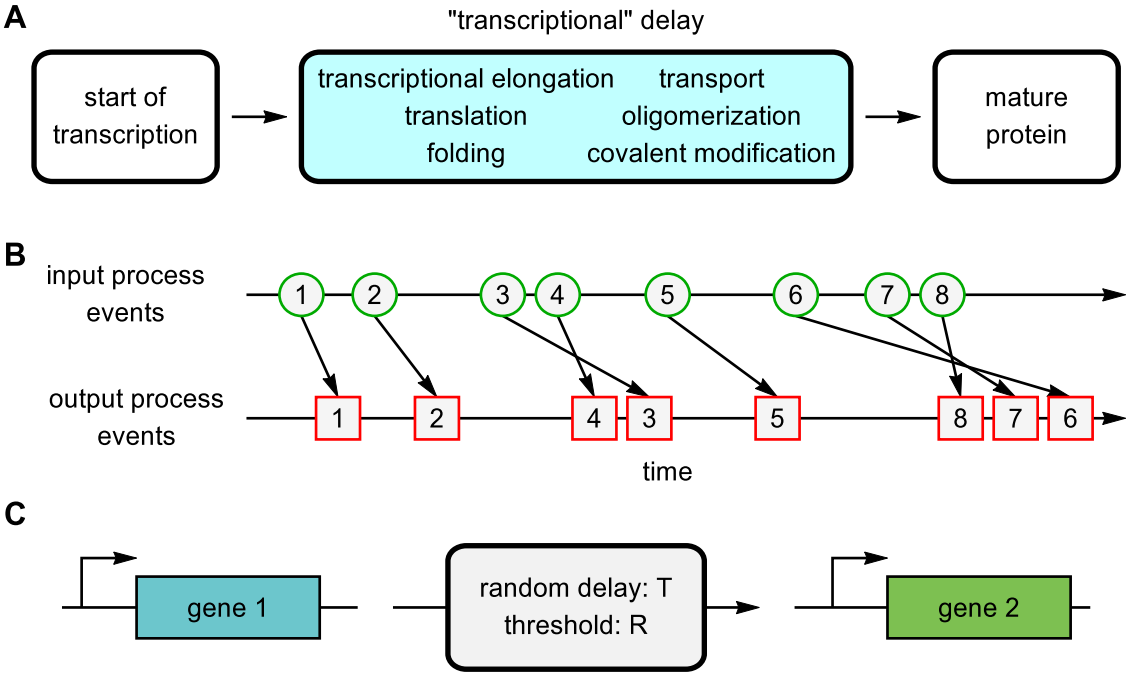

Figure 3.1: **The origin of delay in transcriptional regulation. (A)** Numerous reactions must occur between the time that transcription starts and when the resulting protein molecule is fully formed and mature. Though we call this phenomenon "transcriptional" delay, there are many reactions after transcription (such as translation) which contribute to the overall delay. **(B)** The creation of multiple proteins can be thought of as a queueing process. Nascent proteins enter the queue (an input event) and emerge fully matured (an output event) some time later depending on the distribution of delay times. Because the delay is random, it is possible that the order of proteins entering the queue is not preserved upon exit. **(C)** In a transcriptionally regulated signaling process the time it takes for changes in the expression of gene 1 to propagate to gene 2 depends on both the distribution of delay times, f_T, and the number of transcription factors needed to overcome the threshold of gene 2, R.

In one recent study, Bel *et al.* studied completion time distributions associated with

Markov chains modeling linear chemical reaction pathways [8]. Using rigorous analysis and numerical simulations they show that, if the number of reactions is large, completion time distributions for an idealized class of models exhibit a sharp transition in the coefficient of variation (CV, defined as the standard deviation divided by the mean of the distribution), going from near 0 (indicating a nearly deterministic completion time) to near 1 (indicating an exponentially distributed completion time) as system bias moves from forward to reverse.

However, it is possible, and perhaps likely, that the limiting distributions described by Bel *et al.* do not provide good approximations for protein production. For instance, when the number of rate-limiting reactions is small, but greater than one, the distribution of delay times can be more complex. Moreover, linear reaction pathways only represent one possible and necessarily simplified reaction scheme. Protein production involves many reaction types that are nonlinear and/or reversible, each of which is influenced by intrinsic and extrinsic noise [22], and these reactions may impact the delay time distribution in complicated ways. Therefore, we do not try to derive the actual shape of f_T, but examine the effects its statistical properties have on transcriptional signaling. To do this, we represent protein production as a delayed reaction of the form

$$g \xrightarrow{\lambda(t,P),T} g + P, \tag{3.2}$$

where g is the gene, and transcription is initiated at rate $\lambda(t, P)$, which can depend explicitly on both time and protein number, P. After initiation, it takes a random time, T, for a protein to be formed. Note that the presence of time delay implies that scheme (3.2) defines a non-Markovian process. Such processes can be simulated exactly using the extension of the Gillespie algorithm presented in the previous chapter.

If the biochemical reaction pathway that leads to functional protein is known and relatively simple, direct stochastic simulation of every step in the network is preferable to simulation based on scheme (3.2). From the point of view of multi-scale modeling, however, paradigm (3.2) is useful when the biochemical reaction network is either extremely complex or poorly mapped, since one needs to know only the statistical properties of T.

3.2 Protein formation as a queueing system

In the setting of scheme (3.2), first assume that $\lambda(t, P)$ does not depend on P, and protein formation is initiated according to a memoryless process with rate $\lambda(t)$. A fully formed protein enters the population a random time T after the initiation of protein formation. We assume that the molecules do not interact while forming; that is, the formation of one protein does not affect that of another. Each protein therefore emerges from an independent reaction channel after a random time. This process is equivalent to an $M/G/\infty$ queue [36], where M indicates a memoryless source (transcription initiation), G a general service time distribution (delay time distribution), and ∞ refers to the number of service channels.

In our model, the order in which initiation events enter a queue is not necessarily preserved. As Fig. 3.1 (B) illustrates, it is possible for the initiation order to be permuted upon exit [72]. The assumption that proteins can "skip ahead" complicates the analysis of transient dynamics of such queues, and is essential in much of the following. While there are steps where such skipping can occur (such as protein folding), there are others for which it cannot. For instance, it is unlikely that one RNA polymerase can skip ahead of another – and similarly for ribosomes during translation off of the same transcript. Therefore, protein skipping may be more relevant in eukaryotes, where transcription and translation

must occur separately, than prokaryotes, where they may occur simultaneously. However, if there is more than one copy of the gene (which is common for plasmid-based synthetic gene networks in *E. coli*), or more than one transcript, some skipping is likely occur. Therefore it is likely that the full results that follow are more relevant for genes of copy number greater than one.

3.3 Downstream transcriptional signaling

One purpose of transcription factors is to propagate signals to downstream target genes. Determining the dynamics and stochasticity of these signaling cascades is of both theoretical and experimental interest [43, 82]. Therefore, we will examine the impact that distributed delay has on simple downstream signaling.

Preliminary information We first derive the signaling time distributions for a single gene that is modeled by an $M/G/\infty$ queue. An $M/G/\infty$ queue is a queueing system consisting of a memoryless arrival process (M) and infinitely many service channels (∞). The service time distribution is general (G) and there exists no maximal system size. Let

1. $(I(t))_{t\geq 0}$ denote the input (arrival) process,
2. $(Q(t))_{t\geq 0}$ denote the queue size process, and
3. $(P(t))_{t\geq 0}$ denote the departure (completion) process.

Thus $I(t)$, $Q(t)$, and $P(t)$ are the numbers of proteins that have entered the queue, are in the queue, and have departed the queue, respectively, at time t. Note that $I(t) = Q(t) + P(t)$ for all $t \geq 0$. Suppose that $(I(t))_{t\geq 0}$ is a nonhomogeneous Poisson process

with rate function $\lambda(t)$. Let T denote the (random) service time and let F_T denote the cumulative distribution function (CDF) of T. This is the amount of time that a protein spends in the queue after entering. If the distribution of T is absolutely continuous, let f_T denote the probability density function (PDF) of T. For $t \geq 0$, define

$$\Lambda(t) = \int_0^t \lambda(s)\, \mathrm{d}s.$$

Notice that $E[I(t)] = \Lambda(t)$ for all $t \geq 0$.

Proposition. (transient distributions; see *e.g* [36]) *Let $t \geq 0$. The random variables $Q(t)$ and $P(t)$ are Poisson with means*

$$E[Q(t)] = \Lambda(t) - \int_0^t F_T(s)\,\lambda(s)\, \mathrm{d}s,$$
$$E[P(t)] = \int_0^t F_T(s)\,\lambda(s)\, \mathrm{d}s.$$

Signaling time distributions Let S_R denote the (random) first time at which $P(S_R) = R$, i.e. $S_R = \min\{t : P(t) = R\}$. We rescale time so that the rescaled completion process is a homogeneous Poisson process with rate 1. For $t \geq 0$ define

$$\bar{P}(t) := E[P(t)] = \int_0^t F_T(s)\,\lambda(s)\, \mathrm{d}s.$$

Define the rescaled departure process $\left(\tilde{P}(\zeta)\right)$ by $\tilde{P}(\zeta) = P\left(\bar{P}^{-1}(\zeta)\right)$. Let ξ_R denote the (random) time at which $\tilde{P}(\xi_R) = R$. The random time ξ_R has a gamma distribution with PDF

$$g_{\xi_R}(\zeta) = \frac{\zeta^{R-1}}{(R-1)!} e^{-\zeta}, \tag{3.3}$$

so S_R has PDF

$$f_{S_R}(t) = \frac{\left(\bar{P}(t)\right)^{R-1}}{(R-1)!} e^{-\bar{P}(t)} \bar{P}'(t). \tag{3.4}$$

Computing the expectation of S_R, we have

$$E[S_R] = \int_0^\infty t \left(\frac{\left(\bar{P}(t)\right)^{R-1}}{(R-1)!} e^{-\bar{P}(t)} \bar{P}'(t) \right) \mathrm{d}t \tag{3.5a}$$

$$= \int_0^{\bar{P}(\infty)} \bar{P}^{-1}(\zeta) \left(\frac{\zeta^{R-1}}{(R-1)!} e^{-\zeta} \right) \mathrm{d}\zeta. \tag{3.5b}$$

We now show that if T is symmetrically distributed about its mean and λ is a constant function, then for every fixed value of R, increasing the standard deviation σ_T of T *decreases* the expected signaling time.

Proposition *Assume that T is symmetrically distributed about its mean and that λ is a constant function. Let $R \in \mathbb{N}$. The function $E[S_R]$ is a decreasing function of σ_T.*

Proof. Suppose that $\lambda \equiv \lambda_0$. In light of (3.5b), it suffices to show that for every fixed $t \geq 0$, $\bar{P}(t)$ is an increasing function of σ_T. We write $\bar{P}(t, \sigma_T)$ and $F_T(t, \sigma_T)$ to explicitly indicate the dependence of $\bar{P}$ and F_T on σ_T as well as t. Fix $t \geq 0$ and let $\gamma_2 > \gamma_1$. Define $\tau = E[T]$. For every $0 < z \leq \tau$, we have

$$F_T(\tau - z, \gamma_2) - F_T(\tau - z, \gamma_1) = F_T(\tau + z, \gamma_1) - F_T(\tau + z, \gamma_2) \geq 0.$$

Therefore, if $t < \tau$, we have

$$\bar{P}(t, \gamma_2) = \lambda_0 \int_0^t F_T(s, \gamma_2)\, \mathrm{d}s \geq \lambda_0 \int_0^t F_T(s, \gamma_1)\, \mathrm{d}s = \bar{P}(t, \gamma_1).$$

If $\tau < t < 2\tau$, we have

$$\begin{aligned} \bar{P}(t, \gamma_2) &= \lambda_0 \int_0^{2\tau - t} F_T(s, \gamma_2)\, \mathrm{d}s + \lambda_0 \int_{2\tau - t}^t F_T(s, \gamma_2)\, \mathrm{d}s \\ &\geq \lambda_0 \int_0^{2\tau - t} F_T(s, \gamma_1)\, \mathrm{d}s + \lambda_0 \int_{2\tau - t}^t F_T(s, \gamma_2)\, \mathrm{d}s \\ &= \lambda_0 \int_0^{2\tau - t} F_T(s, \gamma_1)\, \mathrm{d}s + \lambda_0 \int_{2\tau - t}^t F_T(s, \gamma_1)\, \mathrm{d}s \\ &= \bar{P}(t, \gamma_1). \end{aligned} \tag{3.6}$$

Finally, if $t > 2\tau$, then the inequality $\bar{P}(t, \gamma_2) \geq \bar{P}(t, \gamma_1)$ follows from computation (3.6) and the fact that for $s > 2\tau$, $F_T(s, \sigma_T) = 1$ for all relevant values of σ_T. □

Expected value of $Q(S_R)$ Computing the expectation of $Q(S_R)$, we have

$$\begin{aligned} E[Q(S_R)] &= E[E[Q(S_R)|S_R = t]] \\ &= \int_0^\infty \left(\int_0^t \lambda(s)(1 - F_T(s))\, \mathrm{d}s \right) \frac{\bar{P}(t)^{R-1}}{(R-1)!} e^{-\bar{P}(t)} \bar{P}'(t)\, \mathrm{d}t \\ &= \int_0^{\bar{P}(\infty)} \left(\int_0^{\bar{P}^{-1}(\zeta)} \lambda(s)(1 - F_T(s))\, \mathrm{d}s \right) \frac{\zeta^{R-1}}{(R-1)!} e^{-\zeta}\, \mathrm{d}\zeta. \end{aligned}$$

Impact of distributed delay Consider the situation depicted in Figure 3.1 (C), in which the product of the first gene regulates the transcription of a second gene. Using the same nomenclature as in scheme (3.2) we write

$$g_1 \xrightarrow{\lambda(t), T_1} g_1 + P_1 \tag{3.7a}$$

$$g_2 \xrightarrow{h(P_1), T_2} g_2 + P_2, \tag{3.7b}$$

where g_1 and g_2 are the copy numbers of the upstream and downstream genes, P_1 and P_2 are the number of functional proteins of each type, and T_i is the random delay time of gene $i = 1, 2$. The transcription rate of gene 2 depends on P_1 and is given by a Hill function $h(P_1)$. We consider the case in which P_1 activates g_2 (depicted in Figure 3.1) and the case in which P_1 represses g_2.

We now ask: If P_1 starts at zero and gene 1 is suddenly turned on, how long does it take until the signal is detected by gene 2? In other words, assume $\lambda(t) = \lambda_0 \Theta(t)$, where $\Theta(t)$ is the Heaviside step function. At what time t does $P_1(t)$ reach a level that is detectable by gene 2? In order to make the problem tractable, we assume that the Hill

function is steep and switch-like, so that we can make the approximation

$$h(P_1) = h_0\Theta(R - P_1) \qquad \text{(repressor case)}; \tag{3.8a}$$

$$h(P_1) = h_0\Theta(P_1 - R) \qquad \text{(activator case)}. \tag{3.8b}$$

Here h_0 is the maximum transcriptional initiation rate of g_2 and $R > 0$ is the threshold value of the Hill function, *i.e.* the number of molecules of P_1 needed for half repression (or half activation) of gene 2. The second gene therefore becomes repressed (or activated) at the time S_R at which R copies of protein P_1 have been fully formed.

We first examine reaction (3.7a). Assume that at time $t = 0$ there are no proteins in the system. Let $I_1(t)$ denote the number of transcription initiation events that have occurred by time t (the arrival process of the queueing system), $Q_1(t)$ the number of proteins being formed at time t (the size of the queue at time t), and $P_1(t)$ the number of functional proteins that have been completed by time t (the exit process of the queueing system). Since the arrival process is memoryless, $(I_1(t))_{t\geqslant 0}$ is a Poisson process with constant rate λ_0 for $t \geqslant 0$. Hence, the expected value of $I_1(t)$ is $E[I_1(t)] = \lambda_0 t\Theta(t)$.

The exit process, *i.e.*, the number of fully functional proteins that have emerged from the queue, $P_1(t)$, is a nonhomogenous Poisson process with time-dependent rate $\lambda_{P_1}(t) = \lambda_0 F_{T_1}(t)$, where $F_{T_1}(t)$ is the cumulative distribution function (CDF) of the delay time T_1. It then follows that

$$\overline{P}_1(t) \equiv E[P_1(t)] = \lambda_0 \int_0^t F_{T_1}(s)\, \mathrm{d}s.$$

Inactivation (or activation) of gene 2 occurs when enough protein P_1 has accumulated to trigger a transcriptional change, according to Eq. (3.8a) or (3.8b). In other words, the random time it takes for the signal to propagate, S_R, is given by $S_R = \min\{t | P_1(t) = R\}$. Trivially, S_R changes by an amount identical to a change in the mean of the delay

distribution. To examine the effects of randomness in delay on the signaling time, we therefore keep the mean of the delay distribution fixed, $E[T_1] = \tau$, and vary $\mathrm{Var}[T_1]$.

Using equation (3.4), the probability density function of S_R is given by

$$f_{S_R}(t) = \frac{\overline{P}_1(t)^{R-1}}{(R-1)!} e^{-\overline{P}_1(t)} \frac{d\overline{P}_1}{dt}. \tag{3.9}$$

Consequently, the mean and variance of the time it takes for the original signal to propagate to the downstream gene can be written as:

$$E[S_R] = \int_0^\infty \frac{s^{R-1} e^{-s}}{(R-1)!} \overline{P}_1^{-1}(s)\, \mathrm{d}s, \tag{3.10}$$

$$\mathrm{Var}[S_R] = \int_0^\infty \frac{s^{R-1} e^{-s}}{(R-1)!} \left(\overline{P}_1^{-1}(s)\right)^2 \mathrm{d}s - E[S_R]^2. \tag{3.11}$$

To gain insight into the behaviors of Eqs. (3.10) and (3.11), we first examine a representative, analytically tractable example.

Example 1 - Bernoulli delay distributions. Assume that the delay time can take on 2 discrete values, $\tau + a$ and $\tau - a$ with equal probability. That is, suppose that the rate function of the input process is constant and equal to λ_0, and T is a Bernoulli random variable described by the probability measure

$$\beta \delta_{\tau - 2a(1-\beta)} + (1-\beta)\, \delta_{\tau + 2a\beta},$$

where $\beta = 1/2$. We begin by computing $E[S_R]$. Let $x_1 = \tau - 2a(1-\beta)$ and $x_2 = \tau + 2a\beta$. The CDF of T is given by

$$F_T(t) = \begin{cases} 0, & t < x_1; \\ \beta, & x_1 \leqslant t < x_2; \\ 1, & t \geq x_2. \end{cases}$$

For $t \geq 0$ we have

$$\bar{P}(t) = \int_0^t \lambda_0 F_T(s)\, \mathrm{d}s = \begin{cases} 0, & t < x_1; \\ \lambda_0 \beta (t - x_1), & x_1 \leq t < x_2; \\ \lambda_0 (t - \tau), & t \geq x_2. \end{cases}$$

The signaling time S_R has PDF

$$\begin{aligned} f_{S_R}(t) &= \frac{\bar{P}(t)^{R-1}}{(R-1)!} e^{-\bar{P}(t)} \bar{P}'(t) \\ &= \begin{cases} 0, & t < x_1; \\ \dfrac{(\lambda_0 \beta (t - x_1))^{R-1}}{(R-1)!} e^{-\lambda_0 \beta (t - x_1)} \lambda_0 \beta, & x_1 \leqslant t < x_2; \\ \dfrac{(\lambda_0 (t - \tau))^{R-1}}{(R-1)!} e^{-\lambda_0 (t - \tau)} \lambda_0, & t \geq x_2. \end{cases} \end{aligned}$$

Figure 3.2 shows a comparison between these analytical results and stochastic simulations.

We compute $E[S_R]$ using (3.5b). The inverse of $\bar{P}$ is defined only for $t \geq x_1$, thus

$$\bar{P}^{-1}(\zeta) = \begin{cases} \dfrac{\zeta}{\lambda_0 \beta} + x_1, & 0 \leqslant \zeta < \lambda_0 \beta (x_2 - x_1); \\ \dfrac{\zeta}{\lambda_0} + \tau, & \zeta \geq \lambda_0 \beta (x_2 - x_1). \end{cases}$$

Substituting for x_1 and x_2 yields

$$\bar{P}^{-1}(\zeta) = \begin{cases} \dfrac{\zeta}{\lambda_0 \beta} + \tau - 2a(1 - \beta), & 0 \leqslant \zeta < 2\lambda_0 a \beta; \\ \dfrac{\zeta}{\lambda_0} + \tau, & \zeta \geq 2\lambda_0 a \beta. \end{cases}$$

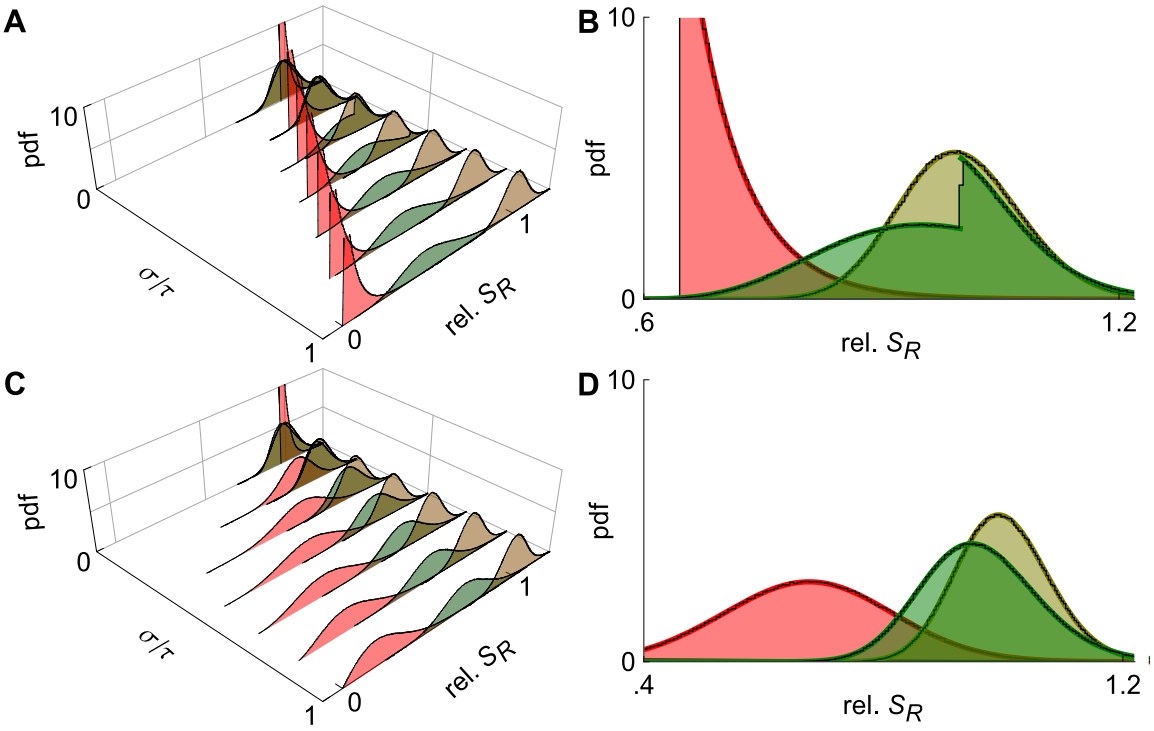

Figure 3.2: **The effects of distributed delay on transcriptional signaling. (A)** PDFs for the signaling time using the delay distribution T from Example 1 with $\beta = 1/2$. The PDFs in red correspond to signal threshold value $R = 1$, green to $R = 10$ and brown to $R = 100$. Here $\tau = 3$ min and $\lambda_0 = 10$ min^{-1}. **(B)** A 2D view of panel (A) with $\sigma = 1$ min. Solid lines show analytical results which are nearly indistinguishable from those obtained through stochastic simulation (black lines). Note that the discontinuity in the green curve is due to the discrete nature of the Bernoulli delay distribution. The CDF, F_T, has jump discontinuities that, in light of Eq. (3.4), produce jump discontinuities in the signaling time PDF. The discontinuity is apparent in both the theoretical prediction (green line) and the stochastic simulations (black line). Panels (C) and (D) are equivalent to panels (A) and (B) with T following a gamma distribution. The PDFs were discretized over 200 bins using 10^6 trials.

Using (3.3) and (3.5b), we have

$$\begin{aligned} E[S_R] &= \int_0^{2\lambda_0 a\beta} \left(\frac{\zeta}{\lambda_0 \beta} + \tau - 2a\,(1-\beta) \right) g_{\xi_R}(\zeta)\, \mathrm{d}\zeta \\ &\quad + \int_{2\lambda_0 a\beta}^{\infty} \left(\frac{\zeta}{\lambda_0} + \tau \right) g_{\xi_R}(\zeta)\, \mathrm{d}\zeta \\ &= \int_0^{2\lambda_0 a\beta} \left(\frac{\zeta}{\lambda_0 \beta} - 2a\,(1-\beta) \right) g_{\xi_R}(\zeta)\, \mathrm{d}\zeta \\ &\quad + \int_{2\lambda_0 a\beta}^{\infty} \frac{\zeta}{\lambda_0} g_{\xi_R}(\zeta)\, \mathrm{d}\zeta \\ &\quad + \tau \int_0^{\infty} g_{\xi_R}(\zeta)\, \mathrm{d}\zeta. \end{aligned}$$

Adding and subtracting $\int_0^{2\lambda_0 a\beta} (\zeta/\lambda_0)\, g_{\xi_R}(\zeta)\, \mathrm{d}\zeta$ gives

$$\begin{aligned} E[S_R] = &\int_0^{2\lambda_0 a\beta} \left(\frac{\zeta}{\lambda_0 \beta} - 2a(1-\beta) \right) g_{\xi_R}(\zeta)\, \mathrm{d}\zeta \\ &- \int_0^{2\lambda_0 a\beta} \frac{\zeta}{\lambda_0} g_{\xi_R}(\zeta)\, \mathrm{d}\zeta \\ &+ \int_0^{2\lambda_0 a\beta} \frac{\zeta}{\lambda_0} g_{\xi_R}(\zeta)\, \mathrm{d}\zeta \\ &+ \int_{2\lambda_0 a\beta}^{\infty} \frac{\zeta}{\lambda_0} g_{\xi_R}(\zeta)\, \mathrm{d}\zeta + \tau \end{aligned}$$

so that we may use $(\zeta/\lambda_0)\, g_{\xi_R}(\zeta) = (R/\lambda_0)\, g_{\xi_{R+1}}$ to obtain

$$E[S_R] = \tau + \frac{R}{\lambda_0} + \int_0^{2\lambda_0 a\beta} \left(\left(\frac{\zeta}{\lambda_0} \right) \left(\frac{1-\beta}{\beta} \right) - 2a(1-\beta) \right) \frac{\zeta^{R-1}}{(R-1)!} e^{-\zeta}\, \mathrm{d}\zeta. \tag{3.12}$$

Finally, we express (3.12) using the upper incomplete gamma function:

$$E[S_R] = \frac{(\beta-1)\left(\Gamma(R+1, 2a\beta\lambda_0) - 2a\beta\lambda_0 \Gamma(R, 2a\beta\lambda_0)\right)}{\beta\lambda_0 \Gamma(R)} + 2a(\beta-1) + \frac{R}{\beta\lambda_0} + \tau, \tag{3.13}$$

where $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, \mathrm{d}t$ and $\Gamma(z,a) = \int_a^\infty t^{z-1} e^{-t}\, \mathrm{d}t$. Since $\beta = 1/2$ we can simplify the mean to

$$E[S_R] = \tau - a + \frac{2R}{\lambda_0} + \frac{a\lambda_0 \Gamma[R, a\lambda_0] - \Gamma[R+1, a\lambda_0]}{\lambda_0 (R-1)!}. \tag{3.14}$$

We now examine the asymptotics of $E[S_R]$ in the $a \to 0$ limit. The first and second partial derivatives of $E[S_R]$ with respect to a are given by

$$\begin{aligned} \partial_a E[S_R] &= \frac{2(\beta-1)\left(\Gamma(R) - \Gamma(R, 2a\beta\lambda_0)\right)}{\Gamma(R)}, \\ \partial_a^2 E[S_R] &= \frac{2^{R+1}(\beta-1) e^{-2a\beta\lambda_0} (a\beta\lambda_0)^R}{a\Gamma(R)}. \end{aligned}$$

Expanding $E[S_R]$ about $a = 0$ gives

$$E[S_R] \sim \tau + \frac{R}{\lambda_0} + \frac{2^{R+1}(\beta-1)(\beta\lambda_0)^R a^{R+1}}{\Gamma(R+2)}.$$

which can be simplified by taking the limit as a goes zero to obtain

$$E[S_R] \approx \tau + \frac{R}{\lambda_0}, \tag{3.15}$$

which is the deterministic limit. The first term is the mean delay time and the second is the average time to initiate R proteins at rate λ_0.

Using the Stirling approximation $n! \sim n^n e^{-n}\sqrt{2\pi n}$ we therefore obtain

$$\frac{2a\left(\beta-1\right)\left(2\lambda_0 a\beta\right)^R}{\left(R+1\right)R!} \sim \frac{2a\left(\beta-1\right)}{\left(R+1\right)\sqrt{2\pi R}}\left(\frac{2\lambda_0 a\beta e}{R}\right)^R,$$

and therefore

$$E[S_R] \sim \tau + \frac{R}{\lambda_0} + \frac{2a\left(\beta-1\right)}{\left(R+1\right)\sqrt{2\pi R}}\left(\frac{2\lambda_0 a\beta e}{R}\right)^R.$$

In particular, for $\beta = 1/2$ we have

$$E[S_R] \sim \tau + \frac{R}{\lambda_0} - \frac{a}{\left(R+1\right)\sqrt{2\pi R}}\left(\frac{\lambda_0 a e}{R}\right)^R.$$

In this case the correction to the deterministic limit is of order $a^{R+1}/R^{R+3/2}$.

We obtain linear large a asymptotics by noting that the first term on the right side of (3.13) vanishes in the $a \to \infty$ limit:

$$E[S_R] \sim 2a\left(\beta-1\right) + \frac{R}{\beta\lambda_0} + \tau.$$

Since $\beta = 1/2$ we obtain

$$E[S_R] \approx \tau + \frac{R}{\lambda_0} - \left(a - \frac{R}{\lambda_0}\right). \tag{3.16}$$

which can be seen in panel (C) in Fig. 3.3. The asymptotic analysis can also be done with other delay distributions to obtain a similar result as in the next example.

Example 2 - Normal delay distributions. Suppose that the rate function of the input process is constant and equal to λ_0. Suppose that T is a normal random variable with mean τ and standard deviation σ.

The CDF of T is given by

$$F_T\left(t\right) = \frac{1}{2}\left(\mathrm{erf}\left(\frac{t-\tau}{\sqrt{2}\sigma}\right)+1\right)$$

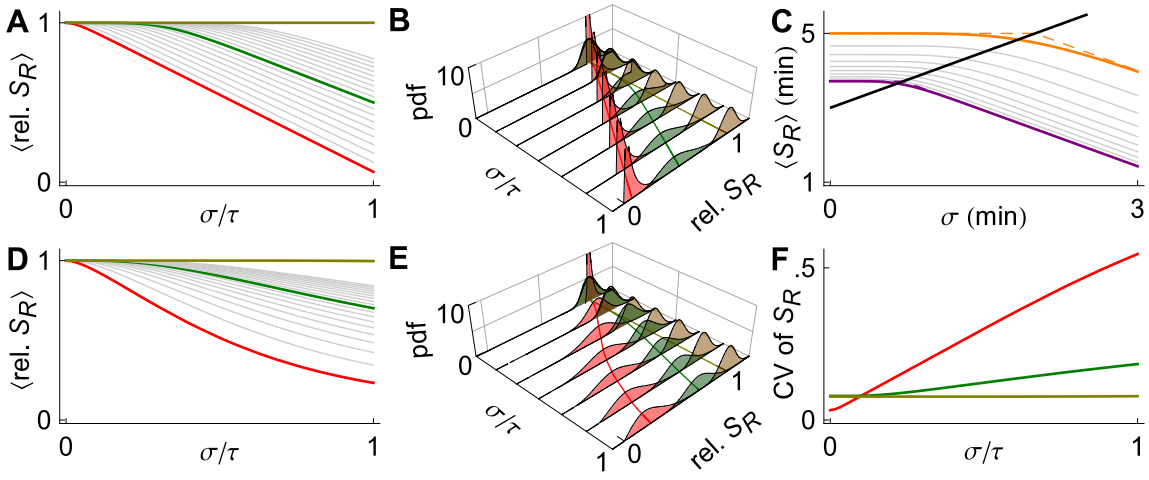

Figure 3.3: **The effects of distributed delay on transcriptional signaling. (A)** For the simplified symmetric distribution where the delay takes values $\tau + a$ and $\tau - a$ with equal probability, the mean signaling time S_R decreases with increasing variability in delay time, Eq. (3.14). Shown are the signaling times (normalized by the time at $\sigma/\tau = 0$), versus CV of the delay time for signaling threshold values from $R = 1$ (red), through $R = 10$ (green) to $R = 19$ in steps of 1. Here $\tau = 3$ min and $\lambda = 10$ min^{-1}. When $R = 100$ (brown) increasing randomness in delay time has little effect on the mean. **(B)** Same as panel (A) but with the probability distribution, f_{S_R}, for different values of σ/τ. **(C)** The transition from the small σ regime to the large σ regime occurs when $\sigma \approx R/\lambda$. Here we fix $R = 10$ and between the different curves vary λ from 5 min^{-1} (magenta) to 14 min^{-1} (orange) in steps of 1. Dashed lines show the asymptotic approximations, Eqs. (3.15) and (3.16), which meet at the black line. Panels **(D)** and **(E)** are equivalent to panels (A) and (B), with T following a gamma distribution, $\tau = 3$ min, and $\lambda = 10$ min^{-1}. **(F)** The coefficient of variation of the signaling time, S_R, as a function of σ.

where erf is the error function. For $t \geq 0$ we have

$$\bar{P}(t) = \frac{1}{2}\lambda_0 \Bigg((t-\tau)\,\mathrm{erf}\left(\frac{t-\tau}{\sqrt{2}\sigma}\right) - \tau \cdot \mathrm{erf}\left(\frac{\tau}{\sqrt{2}\sigma}\right) + \sqrt{\frac{2}{\pi}}\sigma\left(\exp\left(-\frac{(t-\tau)^2}{2\sigma^2}\right) - \exp\left(-\frac{\tau^2}{2\sigma^2}\right)\right) + t \Bigg).$$

Expanding $\bar{P}(t)$ about $\sigma = 0$ we obtain

$$\bar{P}(t) = \begin{cases} 0 + \mathcal{O}(\sigma^3)\left(\exp\left(-\frac{(t-\tau)^2}{2\sigma^2}\right) + \exp\left(-\frac{\tau^2}{2\sigma^2}\right)\right), & t < \tau; \\ \lambda_0(t-\tau) + \mathcal{O}(\sigma^3)\left(\exp\left(-\frac{(t-\tau)^2}{2\sigma^2}\right) + \exp\left(-\frac{\tau^2}{2\sigma^2}\right)\right), & t \geq \tau. \end{cases}$$

Note that the corrections to the first terms in the expansions are exponentially small in σ^2 in both regimes. We denote by P^* the approximation for $\bar{P}$ which omits terms exponentially small in σ^2. The signaling time PDF of S_R can then be approximated by

$$f_{S_R}(t) \approx \frac{(P^*(t))^{R-1}}{(R-1)!} e^{-P^*(t)} \frac{\mathrm{d}}{\mathrm{d}t} P^*(t).$$

Using (3.5a), we have

$$\begin{aligned} E[S_R] &\approx \int_0^\infty t \left(\frac{(P^*(t))^{R-1}}{(R-1)!} e^{-P^*(t)} \frac{\mathrm{d}}{\mathrm{d}t} P^*(t) \right) \mathrm{d}t \\ &= \int_\tau^\infty t \left(\frac{\lambda_0 e^{-\lambda_0(t-\tau)} (\lambda_0(t-\tau))^{R-1}}{(R-1)!} \right) \mathrm{d}t \\ &= \frac{R}{\lambda_0} + \tau, \end{aligned}$$

which is again correct up to terms exponentially small in σ^2.

Signaling acceleration Continuing from the results from Example 1 it follows that for larger delay variability, the mean signaling time *decreases* with delay variability (See Fig. 3.3 (A)). Eqs. (3.15) and (3.16) form the asymptotic boundaries for the mean signaling time. The intersection of the two asymptotes at $a = R/\lambda_0$, gives an estimate of when the behavior of the system changes from the deterministic limit (for $a < R/\lambda_0$) to a regime in which increasing the variability decreases the mean signaling time (for $a > R/\lambda_0$). It follows that the deterministic approximation given by Eq. (3.15) is valid in an increasing range, as R grows (See Fig. 3.3 (C)). The asymptotic analysis of Eq. (3.14) shows that the corrections to Eq. (3.15) are approximately of size $(a/R)^R$, and therefore rapidly decrease with R.

The bottom row of Fig. 3.3 shows that these observations hold more generally: When T_1 is gamma distributed the mean time to produce R proteins, $E[S_R]$, is very sensitive to randomness in delay time, but only when R is small to intermediate. As expected, the densities of the times to produce R proteins, f_{S_R}, are approximately normal and independent of the delay distribution when R is large (Middle panels of Fig. 3.3).

We therefore expect that for each fixed threshold R, $E[S_R]$ is a decreasing function of the standard deviation σ_{T_1} of the delay. We have proved this to be true for symmetric delay distributions. Intuitively, this is due to the fact that the order in which proteins enter the queue is not the same as the order in which they exit. Proteins that enter the queue before the Rth protein, but exit after the Rth protein increase S_R, while the opposite is true for proteins that enter the queue after the Rth protein, and exit before it. Since only finitely many proteins enter the queue before the Rth protein, while infinitely many enter after it, the balance favors a decrease in the mean signaling time. Moreover, as delay variability increases, interchanges in exit order become more likely, and this effect becomes more pronounced. We outline the analytical argument: For each fixed time $t \geqslant 0$, $\overline{P}_1(t)$ is an increasing function of σ_{T_1}, hence $\overline{P}_1^{-1}(s)$ is decreasing function of σ_{T_1} for all $s \geqslant 0$. Referring to Eq. (3.10), this implies that $E[S_R]$ is a decreasing function of σ_{T_1} in the symmetric case.

In sum, mean signaling times decrease as delay variability increases (with fixed mean delay). This effect is most significant for small to moderate thresholds. We note that the decrease in mean signaling time phenomenon depends on a sufficient number of proteins entering the queue. If transcription is only active long enough for less than $2R-1$ proteins to be initiated, then mean signaling time will actually *increase* as delay variability increases. This phenomenon is explained in the next section that analyzes repressor

switches.

3.4 Feedforward loops

Using the above results, we now examine more complicated transcriptional signaling networks. In particular, we turn to two common feedforward loops - the type 1 coherent and the type 1 incoherent feedforward loops (FFL) [58], shown in Fig. 3.4. Each of these networks is a transcriptional cascade resulting in the specific response of the output, gene 3. The coherent FFL generally acts as a delayed response network, while the incoherent FFL has various possible responses, such as pulsatile response [58], response time acceleration [59], and fold-change detection [32].

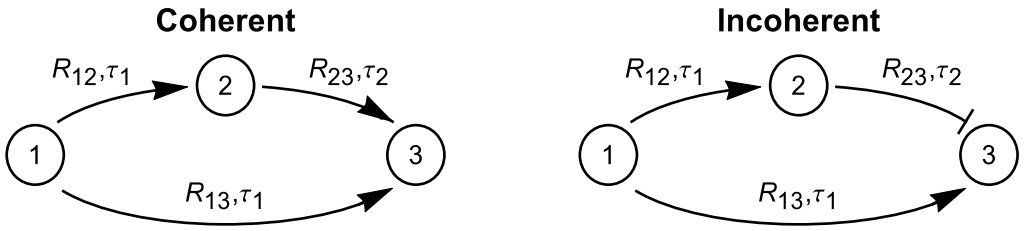

Figure 3.4: **Network schematics for the coherent and incoherent feedforward loops.** Each pathway in the networks has an associated signaling threshold (R_{ij}) and mean delay time (τ_i). The random time between the initiation of transcription of gene i to the full formation of a total of R_{ij} proteins P_i is denoted S_{ij}, which is an implicit function of R_{ij}.

To examine the effects of distributed delay on these networks we consider a network of two $M/G/\infty$ queues with input processes $(I_1(t))_{t\geq 0}$ and $(I_2(t))_{t\geq 0}$, queue size processes $(Q_1(t))_{t\geq 0}$ and $(Q_2(t))_{t\geq 0}$, and departure processes $(P_1(t))_{t\geq 0}$ and $(P_2(t))_{t\geq 0}$. We assume that at $t = 0$ gene g_1 starts transcription of protein P_1 at rate β_1, *i.e* $\lambda_1(t) = \beta_1\Theta(t)$. The second gene, g_2, starts transcription after $P_1(t)$ reaches the threshold R_{12}, so that

$\lambda_2(t) = \beta_2 \Theta(P_1(t) - R_{12})$. The functions $\lambda_1(\cdot)$ and $\lambda_2(\cdot)$ denote the input rate functions of queues 1 and 2, respectively. Queueing system 1 evolves independently of queueing system 2 and acts as a switch: at a time which depends on the exit process of the first system, the input process I_2 switches on (activator switch) or off (repressor switch).

Activator switches For the coherent FFL, we assume that the promoter of gene g_3 acts as an AND gate so that $\lambda_3(t) = \beta_3 \Theta(P_1(t) - R_{13}) \Theta(P_2(t) - R_{23})$. The signaling time between any two nodes i and j within the network, *i.e.*, the random time between the initiation of transcription of gene i to the formation of a total of R_{ij} proteins P_i is denoted S_{ij}. For each of the three pathways, the PDF of the signaling time is given by Eq. (3.9). In addition, because the random times S_{12} and S_{23} are additive (as are their variances), we can directly calculate the time at which P_2 reaches the threshold of gene 3 as $S_{123} = S_{12} + S_{23}$. Therefore, the random time at which the coherent FFL turns on is simply given by $t_{\text{on}} = \max\{S_{13}, S_{123}\}$. Because $E[S_{13}]$ and $E[S_{123}]$ are decreasing functions of the delay variability, it can be expected that so is $E[t_{\text{on}}]$.

Variances of signaling times propagate additively through linear chains of genes in which each gene up-regulates the next. Let $R_{12}, R_{23} \in \mathbb{N}$ be threshold values for protein 1 acting on promoter 2 and protein 2 acting on promoter 3, respectively. We assume that gene 2 is switched on at time

$$S_{12} := \min\{t \geq 0 : P_1(t) = R_{12}\}.$$

Analogously, let S_{23} denote the length of time between S_{12} and the time at which the P_2 process first reaches level R_{23}. The distributions of S_{12} and S_{23} have PDFs of the form given in (3.4). Since S_{12} and S_{23} are independent, we have

$$\text{Var}[S_{12} + S_{23}] = \text{Var}[S_{12}] + \text{Var}[S_{23}].$$

This argument extends inductively to directed pathways in which the product of each gene activates the subsequent gene in the sequence.

Repressor switches For the incoherent FFL we assume that the promoter of g_3 is active only in the presence of P_1 and absence of P_2, so that we may write

$$\lambda_3(t) = \beta_3 \Theta(P_1(t) - R_{13}) \Theta(R_{23} - P_2(t)).$$

In contrast to the coherent FFL, the dynamics of the pulse generating incoherent FFL are less trivial. Since the repressor (P_2) overrides the activator (P_1), assuming $S_{123} \geq S_{13}$ transcription of g_3 turns on at time S_{13} and turns off at time S_{123}, generating a pulse of duration $Z_{\text{on}} = S_{123} - S_{13}$. Note that $E[Z_{\text{on}}]$ can *increase or decrease* as a function of the standard deviation σ of the delay (see Fig. 3.5 where σ was equal for all pathways).

To see this, write $E[Z_{\text{on}}]$ as follows:

$$E[Z_{\text{on}}] = E[S_{12}] + E[S_{23}] - E[S_{13}]. \tag{3.17}$$

Each of the terms on the right side of Eq. (3.17) is the expected signaling time of a single gene ($g_1 \to g_2$, $g_2 \to g_3$, and $g_1 \to g_3$, respectively). Consequently, $E[Z_{\text{on}}]$ depends on σ as a linear combination of 3 expected signaling time curves of the type pictured in Fig. 3.3. The shapes of these signaling time curves determine the behavior of $E[Z_{\text{on}}]$ as a function of σ. Fig. 3.5 shows that the behavior of the duration of the transcriptional pulse as a function of the delay variability depends on the values of each threshold within the network.

Suppose that I_2 is on until time S_{12}, at which point I_2 switches off. Queueing system 2 now has modified input rate function $\lambda_2 \mathbf{1}_{[0,S_{12}]}$, where $\mathbf{1}_J$ is the characteristic function of

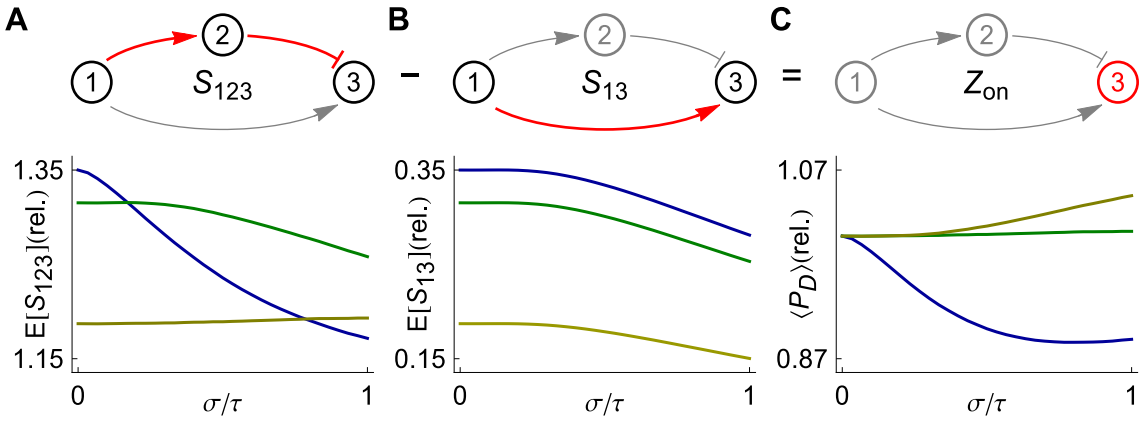

Figure 3.5: **Distributed delay can either increase of decrease pulse duration in an incoherent feedforward loop. (A)** *Top*: The longer pathway consists of the sum of two shorter pathways: $S_{123} = S_{12} + S_{23}$. **(A)** *Bottom*: The expected value of signaling time as a function of the relative standard deviation of the delay time. **(B)** *Top*: The shorter pathway is simply the signaling of the first gene to the third. **(B)** *Bottom*: Expected signaling time, S_{13}. **(C)** *Top*: The output pulse is determined by the amount of time gene 3 is actively transcribing. This time is simply the difference of the longer path duration (S_{123}) and the shorter path duration (S_{13}). **(C)** *Bottom*: Depending on the thresholds R_{12}, R_{13}, and R_{23}, the expected pulse duration can either increase or decrease as a function of the delay variability. In each of the three plots, the data on the vertical axis are presented relative to the mean pulse duration at $\sigma = 0$. Here, the colored lines correspond to $R_{23} = 1$ (blue), $R_{23} = 15$ (green), and $R_{23} = 100$ (brown), while $R_{12} = 100$, $R_{13} = 100$. In addition, the protein degradation rates are each $\beta = (\ln 2)\,/30\ \text{min}^{-1}$, all delays are gamma distributed with mean $\tau = 3$ min.

the interval J. We compute $E[P_2(t)]$ for $t \geq 0$ by conditioning on S_{12}. Let $t \geq 0$. We have

$$E[P_2(t)\,|S_{12} = t^*] = \begin{cases} \int_0^t \lambda_2(s)\, F_{T_2}(s)\ \mathrm{d}s, & \text{if } t \leqslant t^*; \\ \int_0^{t^*} \lambda_2(s)\, F_{T_2}(t - t^* + s)\ \mathrm{d}s, & \text{if } t > t^*. \end{cases} \tag{3.18}$$

Therefore

$$\begin{aligned} E[P_2(t)] &= E[E[P_2(t)\,|S_{12}=t^*]] \\ &= \int_0^\infty E[P_2(t)\,|S_{12}=t^*] f_{S_{12}}(t^*)\ \mathrm{d}t^* \\ &= \int_0^t \left(\int_0^{t^*} \lambda_2(s)\, F_{T_2}(t-t^*+s)\ \mathrm{d}s \right) f_{S_{12}}(t^*)\ \mathrm{d}t^* \\ &\quad + \int_t^\infty \left(\int_0^t \lambda_2(s)\, F_{T_2}(s)\ \mathrm{d}s \right) f_{S_{12}}(t^*)\ \mathrm{d}t^*. \end{aligned}$$

Higher moments may be obtained in a similar manner.

For a repressor switch, the P_2 process and therefore the ability of gene 2 to signal downstream components depend in complex ways on the statistical properties of T_2. We examine these complex relationships by conditioning first on S_{12} and then on I_2. Suppose that $S_{12} = t^*$. The key observation is this: for fixed $t > t^*$, $E[P_2(t)\,|S_{12}=t^*]$ can *increase or decrease* with the standard deviation σ_{T_2} of T_2. We verify this assuming T_2 is symmetrically distributed about its mean and assuming λ_2 is a constant function.

If the midpoint $t - t^*/2$ of the time interval $[t-t^*, t]$ satisfies $t - t^*/2 < E[T_2]$, then

$$\int_0^{t^*} F_{T_2}(t-t^*+s)\ \mathrm{d}s \tag{3.19}$$

is an increasing function of σ_{T_2} and therefore $E[P_2(t)\,|S_{12}=t^*]$ *increases* as σ_{T_2} increases. By contrast, if $t - t^*/2 > E[T_2]$, then the integral in (3.19) is a decreasing function of σ_{T_2} and therefore $E[P_2(t)\,|S_{12}=t^*]$ *decreases* as σ_{T_2} increases. Repressive signaling can therefore qualitatively affect the response of P_2 production to changes in the variability of T_2.

We now examine the ability of gene 2 to signal downstream components by conditioning on I_2. Let $R_{2*} \in \mathbb{N}$. Let S_{2*} denote the time at which P_2 first reaches level R_{2*}. The key observation is this: If we assume that gene 2 shuts off after exactly ζ transcription initiation events, then $E[S_{2*}|\zeta]$ can *increase or decrease* as a function of σ_{T_2}. Fig. 3.6 demonstrates

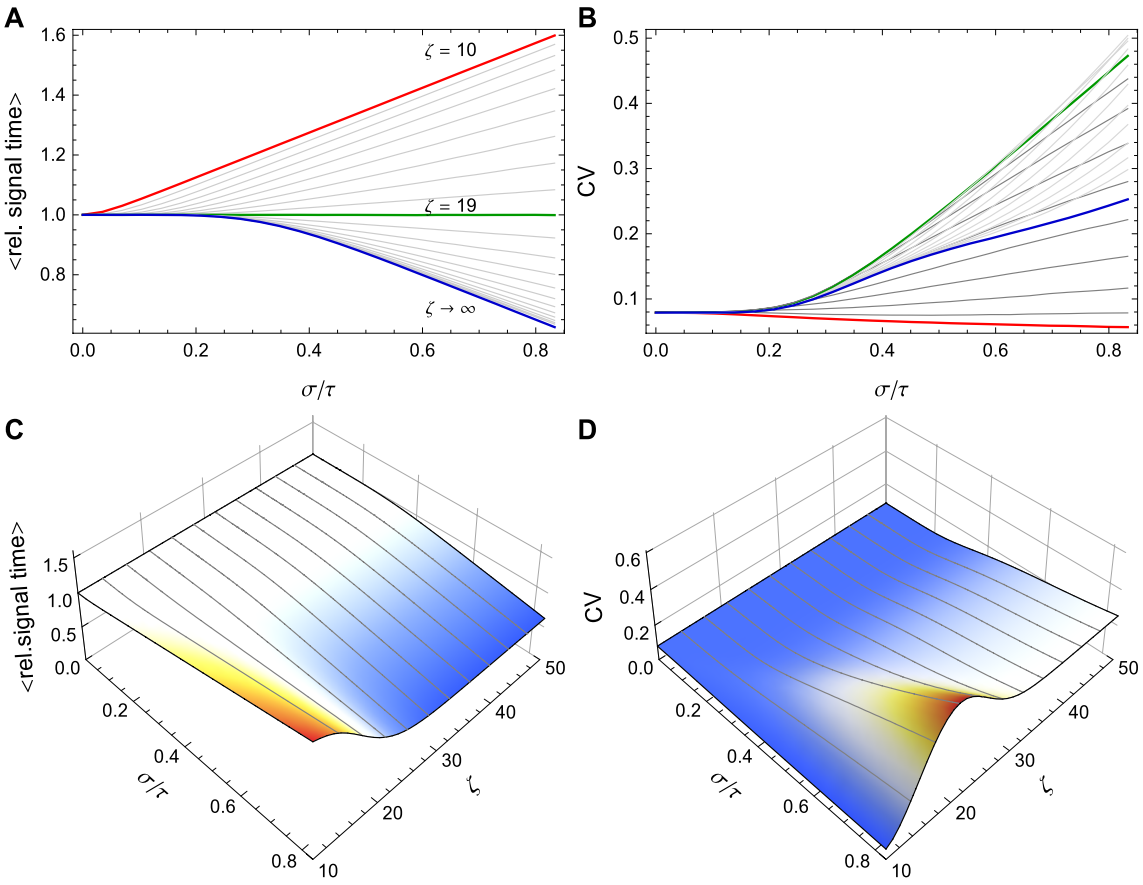

Figure 3.6: **Signaling time depends on the number of initiation events.** $E[S_{2*}|\zeta]$ can *increase or decrease* as a function of σ_{T_2} depending on the value of ζ. Here $R_{2*} = \lambda_2 = 10$. **(A)** $E[S_{2*}|\zeta]$ vs. CV of T_2 for ζ varying from $\zeta = 10$ (red) to $\zeta = 19$ (green) to $\zeta = \infty$ (blue) using the Bernoulli delay distribution T_2 in Example 1 with $\beta = 1/2$. Note the transition that occurs at $\zeta = 19$. **(B)** Equivalent to (A), but plotting CV of the signaling time instead of conditional expectation. (C) and (D) Contour plots corresponding to (A) and (B), respectively. Notice that for fixed σ/τ, signaling time CV can change non-monotonically with ζ. For instance, at $\sigma/\tau = 0.6$, signaling time CV starts low (red), increases to ~ 0.3 (green) and then decreases thereafter. Plots were obtained through stochastic simulation with 10^6 trials.

this numerically for a case in which λ_2 is a constant function and T_2 is symmetrically distributed. In this case, we find that $E[S_{2*}|\zeta]$

1. increases as σ_{T_2} increases if $\zeta < 2R_{2*} - 1$,

2. does not depend on σ_{T_2} if $\zeta = 2R_{2*} - 1$, and

3. decreases as σ_{T_2} increases if $\zeta > 2R_{2*} - 1$.

Intuitively, this is due to the fact that the order in which proteins enter the queue is not necessarily the same as the order in which they exit. Consider again Fig. 3.6. When the total number of transcription initiation events, ζ, is smaller than 19, then more proteins enter the queue before the 10th protein than after it. It is therefore more likely that a protein entering before protein 10 will exit ahead of it than that a protein entering after protein 10 will exit before it. As a result, the expected time $E[S_{2*}|\zeta]$ increases with σ_{T_2}. When the balance favors proteins that enter the queue after protein 10, the opposite is true, and $E[S_{2*}|\zeta]$ decreases with σ_{T_2}.

We conjecture that this trichotomy holds in general if λ_2 is a constant function and T_2 is symmetrically distributed about its mean.

3.5 The delayed negative-feedback oscillator

These observations can also be extended to networks with recurrent architectures. For instance, consider the transcriptional delayed negative feedback circuit [61], which can be described using an extension of scheme (3.2):

$$g \xrightarrow{h(P),T} g + P \tag{3.20a}$$

$$P \xrightarrow{\mu(P)} \emptyset, \tag{3.20b}$$

where $h(P)$ is a decreasing Hill function (i.e., P represses its own production) and $\mu(P)$ is the degradation rate due to dilution and proteolysis. Mather *et al.* examined the oscillations produced by systems of the type described by (3.20) when the delay T is nonrandom (degrade and fire oscillators) [61]. Starting with no proteins, P is produced at a rate governed by the Hill function h. When the level of P exceeds the midpoint of the Hill function, gene g effectively shuts down. The proteins remaining in the queue exit, producing a spike, after which degradation diminishes P. When the protein level drops sufficiently, reaction (3.20a) reactivates and production of P resumes, commencing another oscillation cycle. Note that this circuit will not oscillate without delay.

As a result during each oscillation the gene is turned on until its own signal reaches itself, at which time the gene is turned off [61]. Therefore, the peak height of one oscillation is determined by the length of time the gene was in the "on" state. Since that time is determined by the gene's signaling time, our theory predicts that the mean peak height of the oscillations will decrease as the variability in the delay time increases. Indeed, this is exactly what our stochastic simulations show when the delay time, T, is gamma distributed (Fig. 3.7). This is consistent with the fact that the negative feedback circuit is dynamically similar to the g_2-g_3 sub-circuit within the incoherent FFL.

We can use our theory to predict the change in the peak height of the oscillator as a function of σ. For a delay that is gamma-distributed, the change in signaling time as a function of σ can be written as

$$f(\sigma) = E_{\sigma=0}[S_R] - E_{\sigma}[S_R],$$

where $E_{\sigma}[S_R]$ is given by Eq. (3.10) and $f(\sigma)$ is the reduction in the expected signaling time. If we assume that the amount of time that protein is produced during a burst in the delayed negative feedback oscillator is also reduced by this amount, then it is possible to

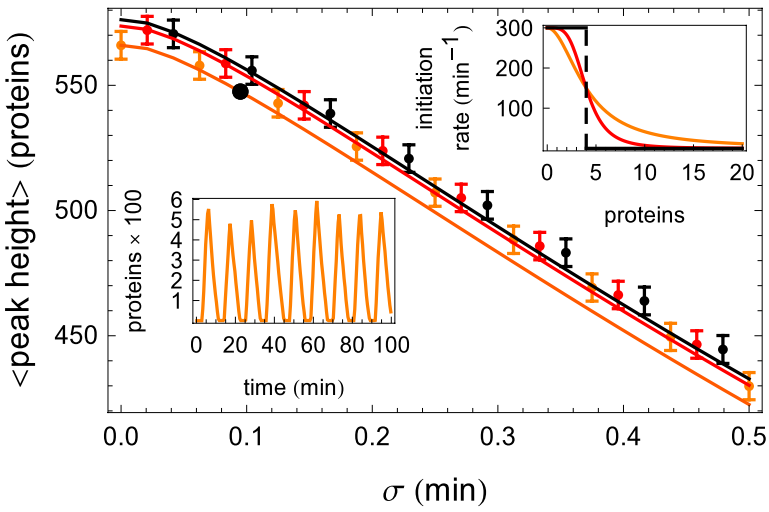

Figure 3.7: **Distributed delay in the delayed negative feedback oscillator.** Shown are the analytically predicted (solid lines) and numerically obtained (symbols with standard deviation error bars) mean peak heights of the negative feedback oscillator with Hill coefficients of $n = 2$ (orange), $n = 4$ (red), and $n = \infty$ (i.e. step function, black). The top inset shows the shape of the Hill function for the three values of n, with colors matching those in the main figure. The lower inset shows one realization of the oscillator at parameter values corresponding to the large black circle on the orange ($n = 2$) curve of the main figure. The average and the standard deviation of the peak heights were calculated from stochastic simulations of 10^5 oscillations. Here $a = 300\,\mathrm{min}^{-1}$, $\beta = 0.1\,\mathrm{min}^{-1}$, $\gamma_R = 80\,\mathrm{min}^{-1}$, $C_0 = 4$ and $R_0 = 1$.

predict the change in the peak height accordingly. To a first approximation, if the promoter is in the "ON" state for a time that is $f(\sigma)$ less, then a total of $\alpha f(\sigma)$ less protein will be produced. Therefore we can write the expected peak height of the oscillator as

$$E_\sigma[PH] \approx E_{\sigma=0}[PH] - \alpha f(\sigma)\,. \tag{3.21}$$

However, due to degradation, Eq. (3.21) overestimates the correction to the peak height. Due to exponential degradation, only a fraction $\exp(-\beta f(\sigma))$ of the lost protein would have made it through to the peak. Also, the duration of enzymatic decay is also reduced by a time $f(\sigma)$. Therefore, if we assume that the enzymatic decay reaction is saturated, we need to add an amount $f(\sigma)\gamma_R$ to Eq. (3.21). This gives us a more accurate prediction

of the mean peak height as

$$E_\sigma[PH] \approx E_{\sigma=0}[PH] - \alpha f(\sigma) e^{-\beta f(\sigma)} + f(\sigma) \gamma_R.$$

Figure 3.7 shows that this approximation works well, even for a Hill coefficient as low as 2.

3.6 Discussion

The existence of delay in the production of protein has been known of for some time. For many systems its presence does not seriously impact performance. For example, the existence of fixed points in simple downstream regulatory networks without feedback is unaffected by delay. Delay is important if the timing of signal propagation impacts the function of the network. Delay can also change a network's dynamics. In networks with feedback, for instance, delay can result in bifurcations that are not present in the corresponding non-delayed system. The delayed negative feedback oscillator is a prime example [61]. Moreover, while the effect of delay in a single reaction may be small, it is cumulative and linearly additive in directed lines.

The intrinsic stochasticity of the reactions that create mature protein make some variation in delay time inevitable. However, we do not yet know the exact nature of this variability or the functional form of the probability density function f_T. To further complicate matters, there may exist a substantial amount of extrinsic variability in the delay time – the statistics of the PDF may vary from cell to cell.

We focused on the transient dynamics of $M/G/\infty$ queues in order to demonstrate the effects of distributed delay in a tractable setting. However, as mentioned earlier, $M/G/\infty$ queues may not always be a good model for protein production. For genes with low

copy number or few available transcripts queues with $L < \infty$ service channels ($M/G/L$ queues) may provide a better description. For eukaryotic systems models in which transcription and translation are decoupled into separate queues may also be relevant. In addition, as protein production rates are often coupled with extrinsic factors such as growth rate and cell cycle phase, L may depend on time and on the state of the system.

The complexity of biochemical reaction networks suggests the use of networks of queues [2], and sources could be toggled on and off by other components of a reaction network. Even protein production from a single transcript may be more accurately described by a sequence of $M/G/1$ queues with each codon as one in a chain of service stations. In such a model ribosomes move from one codon station to the next, and are not able to skip ahead.

One further complication occurs if the burstiness of the promoter is large [33]. In the above analysis, we assumed that the initiation events of proteins were exponentially distributed in time. Since this is not necessarily the case due to the burstiness of promoters, some limits need to be put on the usefulness of the above results. Equations (3.15) and (3.16) suggest that the transition to accelerated behavior occurs when

$$\sigma > R/\lambda. \tag{3.22}$$

One can think of R/λ as the average time, $\overline{T}_R$, it takes to initiate R proteins, and rewrite the boundary as $\sigma > \overline{T}_R$. One can then assume that if the burstiness of the initiation events is not large, *i.e.* that the mean burst size is less than the signal threshold, then it does not matter what the distribution of initiation events is. In other words, as long as approximately R proteins are initiated in the time $\overline{T}_R$, and the variance of that number is not large, then Eq. (3.22) still holds.

Chapter 4

Transcriptional delay in bistable gene networks

Here we investigate a variety of bistable gene networks using the modified version of the Gillespie algorithm that allows us to incorporate transcriptional delay[1]. Although the details and dimensionality of the networks differ, in each the mean residence times near the stable states *increase* dramatically with even modest increases in transcriptional delay time (See Fig. 4.1). In some cases, stability increases despite the fact that the stationary distributions show no appreciable change. To explain this phenomenon, we construct a non-Markovian, analytically tractable model. The model predicts that the enhanced stability is due to an increase in the number of failed transitions between stable states. Exact stochastic simulations of each bistable system (a single-gene positive feedback loop; the co-repressive toggle switch [26]; and the lysis/lysogeny switch of phage λ [95]) verify this prediction.

[1]Content in this Chapter has been previously published: Gupta, C., López, J. M., Ott, W., Josić, K., Bennett, M. R. (2013). **Transcriptional delay stabilizes bistable gene networks**. *Physical review letters*, 111(5), 058104.

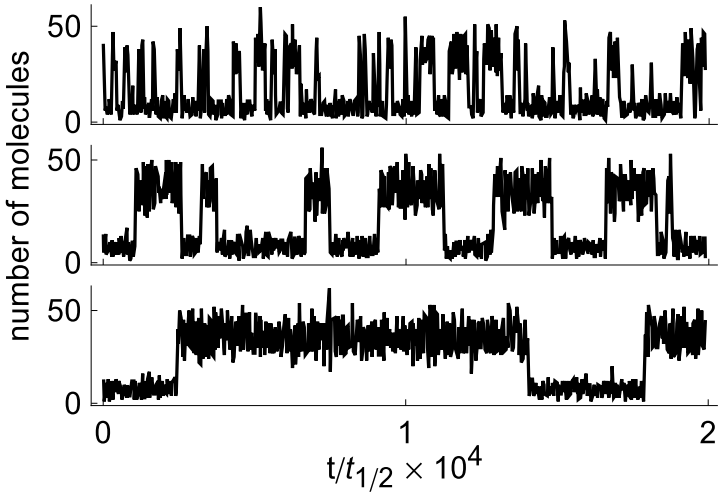

Figure 4.1: **Sample trajectories for a single gene positive feedback loop.** From top to bottom, the three timeseries correspond to transcriptional delays $\tau = 0, t_{1/2}$, and $2t_{1/2}$, where $t_{1/2}$ is the half-life of the protein.

4.1 Reduced Model

Here we present the detailed analysis of the reduced model, and derive the primary expression of interest. In order to obtain a unified description of the observed increase in stability with an increase in transcriptional delay, we introduce a generalized 3-state reduced model (RM). Two of the states in the model correspond to neighborhoods of the two stable fixed points. We call these states H (high) and L (low). The third state is an intermediate state, I, corresponding to a neighborhood of the separatrix. All transitions between H and L must pass through I. Therefore, the RM represents a coarse projection of a general bistable model where large fluctuations push the system from the stable states into a neighborhood of the separatrix.

Due to transcriptional delay, the transition rates between the states depend on the history of the system. This is particularly important when the system is in state I. In the

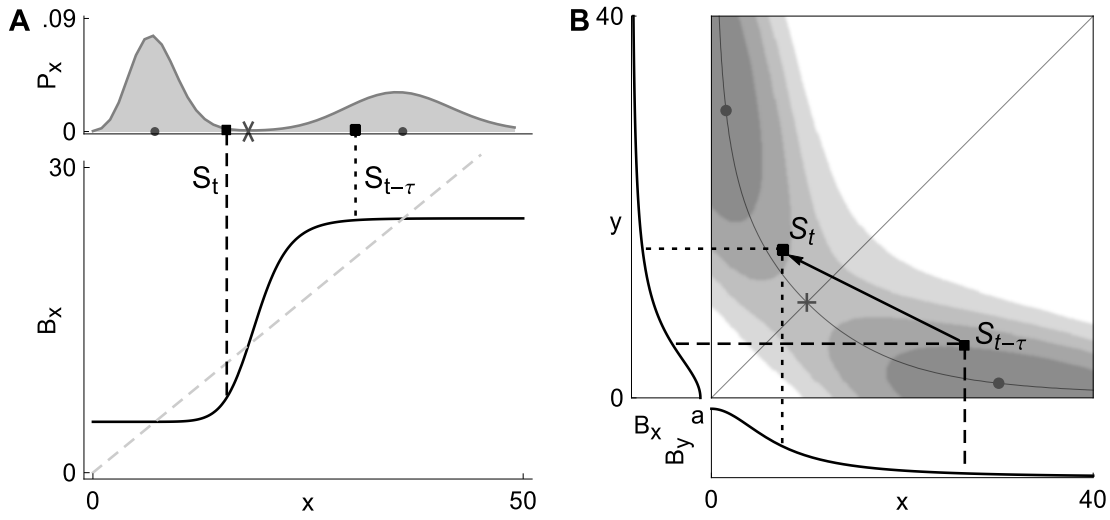

Figure 4.2: **Phase space dynamics.** The state of the system at time $t - \tau$ and time t is shown in the case of **(A)** a positive feedback loop, and **(B)** a genetic toggle switch. Mature proteins enter the population at time t at rates determined by the state at time $t - \tau$. In **(A)**, at time t, the birth rate B_x of x is larger in the past, and production is higher than if the birth rate was determined by the present state of the system, S_t. This facilitates a return to the previously visited stable state. A similar explanation holds for **(B)** (see text). Stationary densities are shown in both cases.

absence of delay the system has no memory; the likelihood of a transition to either stable state from a neighborhood of the separatrix is determined only by the present state of the system. However, in the presence of delay, this likelihood will depend on the past.

As a particular example, consider the positive feedback loop (Fig. 4.3A). At the upper stable state, which corresponds to state H in the RM, protein production is high. Consider a large fluctuation from this state that takes the system away from the upper fixed point, to state S_t shown in Fig. 4.2A. In the presence of transcriptional delay, birth rates are determined by the state $S_{t-\tau} = x\,(t - \tau)$. The larger τ, the more likely it is that $x\,(t - \tau)$ is in state H, near the fixed point whose neighborhood has just been abandoned. But the birth rates in state H are high and favor motion back toward H (See Fig. 4.2A). Therefore,

the trajectory is pulled back towards the stable state it came from. Thus, memory in the system acts as a "rubber band" and causes resistance to transitions.

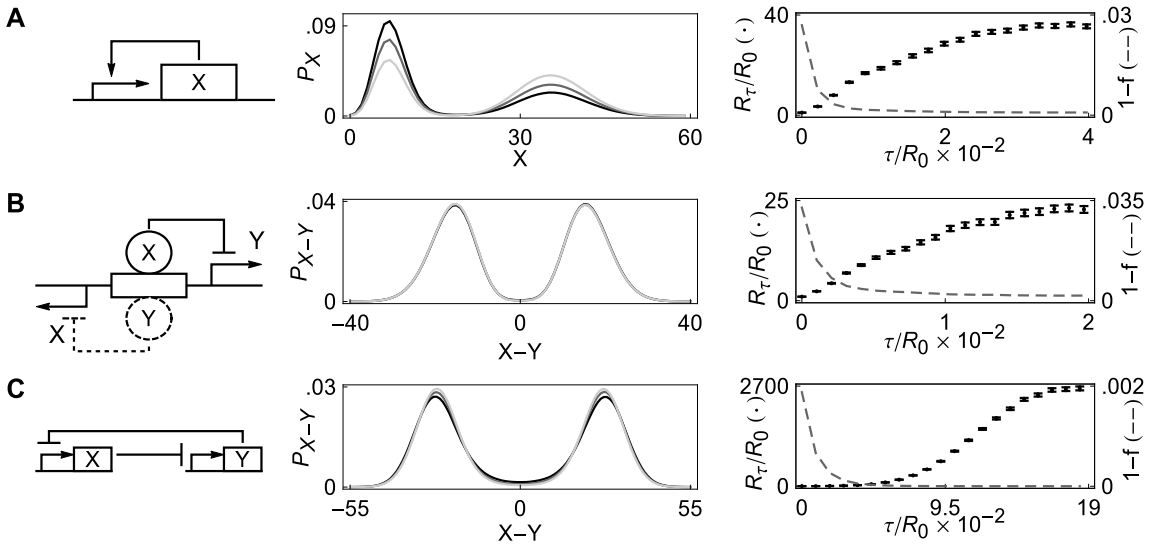

Figure 4.3: **The impact of transcriptional delay on three different genetic networks.** Left panels show the three different gene networks, center panels show the stationary distributions at three different values of transcriptional delay, τ, and right panels illustrate the increase in residence times (dots) and the decrease in the probability of a successful transition (dashed lines) with increasing τ. R_τ denotes the mean residence time in the metastable states at delay τ. The lighter stationary distributions correspond to larger delays; **(A)** Positive feedback model with stationary distributions at $\tau = 0, 1, 2$, and $R_0 = 227$; **(B)** $\lambda-$phage model with stationary distributions at $\tau = 0, 5, 10$, and $R_0 = 4829$; **(C)** Co-repressive toggle switch with stationary distributions at $\tau = 0, 0.45, 0.9$, and $R_0 = 489$.

The situation is similar for the genetic toggle switch, a network of two mutually repressing genes expressing proteins x and y. At the stable states of this system (dots in Fig. 4.2B), the birth rate of one protein is high, and that of the other is low. Consider a fluctuation away from state H, where x is high and y low, to state $S_t = (x(t), y(t))$. Birth rates are again determined by the state $S_{t-\tau} = (x(t-\tau), y(t-\tau))$. As shown in Fig. 4.2B, in state $S_{t-\tau}$ *both* the birth rates of x and y at $t-\tau$ favor motion back to H. Hence in systems of interacting genes, the "rubber band" effect may be even stronger.

To capture the effects of memory in the RM, the transition rates between states are assumed to depend on the state of the system in the past. We define $\lambda^i_{j\to k}$, for $i, j, k \in \{H, I, L\}$, as the rate of transition from state j to state k, given that τ units in the past the system was in state i. Not all transitions are possible, as transitions out of states H and L must go into state I.

We make several assumptions on these transition rates. First, the delay τ is small compared to the mean residence time in each of the stable states. Therefore, if the system is in state H or L at time t, it is unlikely that it was in state I at time $t - \tau$.

Second, we assume that the six transition rates, $\lambda^j_{I\to k}$, out of state I are at least an order of magnitude larger than transition rates out of states L and H. This corresponds to the assumption that the system will exit the vicinity of the separatrix much more quickly than the vicinity of a stable fixed point.

Finally, we assume that for time τ after entering state I, the system is more likely to return to its previous state. In other words, we assume

$$p^H_{I\to H} > p^I_{I\to H}, \qquad p^L_{I\to L} > p^I_{I\to L}, \tag{4.1}$$

where

$$p^i_{I\to j} = \frac{\lambda^i_{I\to j}}{\lambda^i_{I\to H} + \lambda^i_{I\to L}}$$

is the probability of transitioning from state I to state $j \in \{H, L\}$ given the system was in state i a time τ in the past. This assumption captures the "rubber band" effect illustrated in Fig. 4.2, *i.e.* delayed protein production favors a return to the stable state that was visited last.

4.1.1 Metastability as a function of delay

We now analyze the stability of states H and L as a function of the delay, τ. Let R_H denote the residence time for state H. We compute the expected value $E[R_H]$; the computations for L are analogous. Once the system enters state H, it will make a number of failed transitions of the form $H \to I \to H$ before eventually making a successful transition $H \to I \to L$. Let f_H denote the probability of a failed transition, that is, the probability that a transition $H \to I \to H$ occurs conditioned on the system having been in state H for at least time τ. We have

$$f_H = \left(1 - Z_H(\tau)\right) p^H_{I\to H} + Z_H(\tau)\, p^I_{I\to H}, \tag{4.2}$$

where we define $Z_H(\tau)$ by

$$Z_H(\tau) := \exp\left(-\left(\lambda^H_{I\to H} + \lambda^H_{I\to L}\right)\tau\right).$$

Note that f_H is a convex linear combination of $p^H_{I\to H}$ and $p^I_{I\to H}$. When $\tau = 0$ (Markovian case), $f_H = p^I_{I\to H}$. As τ increases away from zero, f_H moves toward $p^H_{I\to H}$.

Let F_H denote the random time needed to complete a failed transition and let S_H denote the time needed for a successful transition. Assuming that the delay is small compared to the characteristic residence times in the stable states, we obtain the key estimate for $E[R_H]$. Writing $R = R_H$, $f = f_H$, $F = F_H$, $S = S_H$, and $\lambda = \lambda^H_{H\to I}$, we have

$$E[R] \sim \frac{f}{1-f}\left(E[F] + \frac{1}{\lambda}\right) + E[S] + \frac{1}{\lambda}. \tag{4.3}$$

The primary contribution in Eq. (4.3) comes from the term $f(1-f)^{-1}$ representing the mean number of failed transitions of the form $H \to I \to H$ before a successful transition $H \to I \to L$. The terms $E[F]$ and $E[S]$ are not very sensitive to τ. On the other hand, because of the inequalities (4.1), $f(1-f)^{-1}$ grows rapidly as τ increases from $\tau = 0$. If

the states H and L are sufficiently stable, the expected time spent in each state before a large fluctuation is approximately λ^{-1}.

Fig. 4.4B illustrates that the RM qualitatively behaves like the models shown in Fig. 4.3.

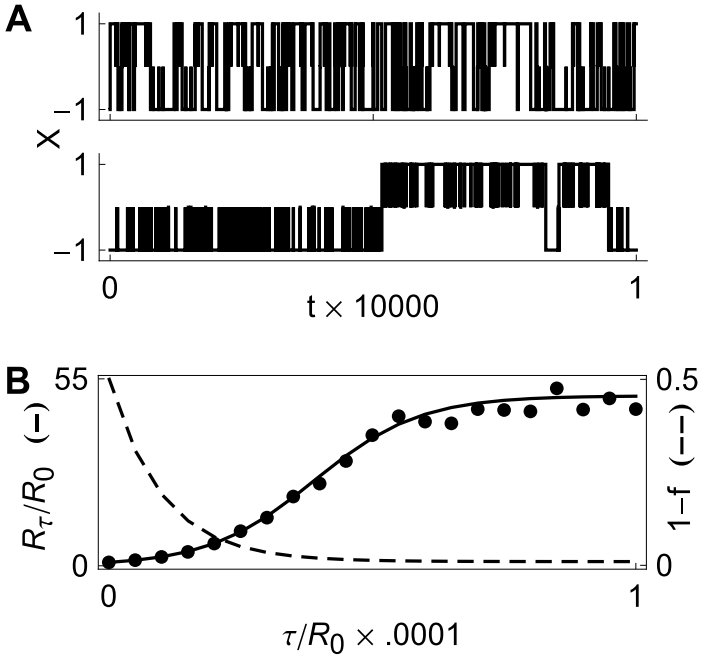

Figure 4.4: **The reduced model. (A)** Sample trajectories for the 3-state RM. Top and bottom trajectories correspond to $\tau = 0$ and $\tau = 0.03$, respectively. **(B)** The estimated increase in mean residence time as a function of delay given by Eq. (4.3) *(Left axis, solid line)* compared to values obtained by simulating the RM (bold circles). The dashed line represents the probability of a successful transition as a function of delay *(Right axis)*.

4.1.2 Analysis

To analyze the RM we first discretize time using a step size Δ and to obtain the probability of making failed transitions for a given delay τ. The probability density function for the random variable that counts the number of failed transitions in the continuous limit is

obtained by taking the limit $\Delta \to 0$. This allows us to compute the mean number of failed transitions and the mean time spent in a failed transition, as well as the mean time spent in a successful transition, assuming that the residence times dominate the delay. These are the primary ingredients required for the computation of the residence times in the stable states.

If the delay is assumed to be $K\Delta$ (where K is some positive integer), the RM can be embedded in a $(K+1)$-dimensional space, and can be represented by vectors $\vec{x} = \left(x_{-K\Delta}, x_{-(K-1)\Delta}, \ldots, x_{-\Delta}, x_0\right)$ with the state x_0 being the current state of the RM and $x_{-t\Delta}$ being the state t steps in the past. In the higher-dimensional space, not all jumps are feasible, and the process can only possibly jump from a configuration $\vec{x}$ to a configuration $\vec{y}$ if $x_{-i\Delta} = y_{-(i+1)\Delta}$ for all $0 \leq i \leq K-1$. The delay can now be expressed by saying that the probability of a feasible jump from $\vec{x}$ to $\vec{y}$ is given by $\Lambda_{x_0 \to y_0}^{x_{-K\Delta}}$. Call f_H the probability of failing a transition, given that the transition starts in the state H, and $x_{-i\Delta} = H$ for all $0 \leq i \leq K$. Let $P(k)$ denote the probability mass function for the number of steps k that are required to complete the loop $H \to I \to H$ given that $H \to I$ has occurred. We see that

$$P(k) = \frac{1}{f_H} \begin{cases} \left(\Lambda_{I\to I}^{H}\right)^{k-1} \left(1 - \Lambda_{I\to I}^{H}\right) \frac{\Lambda_{I\to H}^{H}}{\Lambda_{I\to H}^{H} + \Lambda_{I\to L}^{H}} & 1 \leq k \leq K; \\ \left(\Lambda_{I\to I}^{H}\right)^{K} \left(\Lambda_{I\to I}^{I}\right)^{k-K-1} \left(1 - \Lambda_{I\to I}^{I}\right) \frac{\Lambda_{I\to H}^{I}}{\Lambda_{I\to H}^{I} + \Lambda_{I\to L}^{I}} & k \geq K+1. \end{cases} \tag{4.4}$$

In the continuous-time limit, a discrete delay $K\Delta$ is replaced by a delay τ where $\Delta \to 0$ and $K\Delta \to \tau$. As in the discrete case, for the continuous-time system let f_H denote the probability of failing a transition, given that the transition initiates from state H and the process remembers only state H when the transition begins. Let $P(t)$ denote the probability density function for the random variable F_H: the time needed to complete the $H \to I \to H$ loop given that $H \to I$ has occurred.

	H	**I**	**L**
$\tau = 0$	0.502	2.019×10^{-5}	0.498
$\tau = 0.04$	0.498	1.977×10^{-5}	0.502
$\tau = 0.08$	0.500	2.045×10^{-5}	0.499

Table 4.1: Shown here are the stationary distributions for the RM for three different values of τ: $\tau = 0, \tau = 0.04$ and $\tau = 0.08$. Rates used in the RM are $\lambda^I_{I\to x} = 50, \lambda^x_{I\to x} = 99, \lambda^x_{I\to x^c} = 1, \lambda^I_{x\to I} = 0.20, \lambda^x_{x\to I} = 0.0002, \lambda^x_{x^c\to I} = 0.004; x \in \{H, L\}; x^c = H$ if $x = L$ and $x^c = L$ if $x = H$.

For the continuous-time case, we compute a formal limit in Eq. (4.4). To do so, we set up some notation. For a continuous-time Markov process, a transition probability in a time interval of length Δ is given by $\lambda\Delta$ were λ is the corresponding transition rate for the process. In the discrete-time description of the process, we can replace probabilities such as $\Lambda^j_{i\to k}$ $(i \neq k)$ by $\lambda^j_{i\to k}\Delta$ (rates corresponding to transitions) and probabilities $\Lambda^j_{I\to I}$ by $\left(1 - \left(\lambda^j_{I\to H} + \lambda^j_{I\to L}\right)\Delta\right)$ for $j \in \{H, I, L\}$. For fixed t and any Δ such that $t\Delta^{-1}$ is a positive integer, Eq. (4.4) gives

$$P\left(F_H \in [t-\Delta, t]\right) = \frac{1}{f_H}\begin{cases} \left(1 - \left(\lambda^H_{I\to H} + \lambda^H_{I\to L}\right)\Delta\right)^{t\Delta^{-1}-1}\lambda^H_{I\to H}\Delta & \Delta \leq t \leq \tau \\ \left(1 - \left(\lambda^H_{I\to H} + \lambda^H_{I\to L}\right)\right. & \\ \quad \left.\cdot\,\Delta^{\tau\Delta^{-1}}\left(1 - \left(\lambda^I_{I\to H} + \lambda^I_{I\to L}\right)\Delta\right)^{(t-\tau)\Delta^{-1}-1}\right. & t \geq \tau + \Delta. \\ \quad \cdot\,\lambda^I_{I\to H}\Delta & \end{cases}$$

Taking $\Delta \to 0$, we obtain

$$P\left(t\right) = \frac{1}{f_H}\begin{cases} \lambda^H_{I\to H}\exp\left(-\left(\lambda^H_{I\to H} + \lambda^H_{I\to L}\right)t\right) & 0 < t \leq \tau \\ \lambda^I_{I\to H}\exp\left(-\left(\lambda^H_{I\to H} + \lambda^H_{I\to L}\right)\tau - \left(\lambda^I_{I\to H} + \lambda^I_{I\to L}\right)(t-\tau)\right) & t > \tau \end{cases}. \tag{4.5}$$

Since $P\left(t\right)$ is a pdf, it follows from integrating on $[0, \infty$ that

$$f_H = \left(1 - Z_H\left(\tau\right)\right)p^H_{I\to H} \;+\; Z_H\left(\tau\right)p^I_{I\to H}$$

where

$$p^i_{I\to H} = \frac{\lambda^i_{I\to H}}{\lambda^i_{I\to L} + \lambda^i_{I\to H}}, \quad Z_H(\tau) := \exp\left(-\left(\lambda^H_{I\to H} + \lambda^H_{I\to L}\right)\tau\right).$$

Analogous computations can be performed for the probability of failing a transition if it is assumed that the transition initiates from state L, and the only remembered state is L. Analogous computations can also be performed to obtain the probability density function for the time spent in making a successful transition loop $H \to I \to L$, and, therefore, the probability of a successful transition. Further, we can compute the expectations of F_H, and S_H (defined as the time spent in a successful transition):

$$E[F_H] = \int_0^\infty tP(t)\ dt$$

et cetera. The exact forms of these expressions can be computed analytically, but we do not write them here. Instead, we note that $dE[F_H]/d\tau$ and $dE[S_H]/d\tau$ are very small. This implies that the times spent in state I during a failed transition, or a successful one, do not significantly change with τ.

Denote by N the number of failed transitions before a successful transition. Then N has a geometric distribution ($P(N = n) = f_H^n(1 - f_H), n \geq 0$).

In between each failed transition, the process spends time in the stable states H or L before jumping out on another excursion. We have assumed that the residence times sufficiently dominate the delay; a consequence of this assumption is that once the process re-enters the state H, it stays there long enough to forget its past excursions. Therefore, we can estimate the time between transition attempts by $1/\lambda$ where $\lambda = \lambda^H_{H\to I}$.

We can now estimate the expected residence time in the stable state:

$$E[R_H] \sim \frac{f_H}{1 - f_H}\left(E[F_H] + \frac{1}{\lambda}\right) + E[S_H] + \frac{1}{\lambda}.$$

4.2 Positive Feedback Model

The deterministic delay-differential equation that approximates the stochastic dynamics of the single gene positive feedback model is given by

$$\dot{x} = \alpha + \beta \frac{x\left(t-\tau\right)^{b}}{c^{b} + x\left(t-\tau\right)^{b}} - \gamma x \tag{4.6}$$

The parameters used were $\beta = 20$, $\alpha = 5$, $c = 19$, $\gamma = \ln(2)$ and $b = 10$. The parameter α is the basal rate of production of the molecules of x (with units molecules s^{-1}). The maximal rate of production for the system is $\alpha + \beta$. c is the number of molecules of x required to achieve the half-maximal activation rate of $\alpha + \beta/2$. The constant b is the Hill coefficient for the activation function, and γ is the rate constant for the degradation of the protein x (with units s^{-1}).

For this set of parameters, there are three fixed points for the delayed differential equation: $x = 7.2, x = 18.0$ and $x = 36.0$. The fixed points at 7.2 and 36.0 are stable, while the fixed point at $x = 18.0$ is unstable.

We map the phase space of the positive feedback model onto the RM using $H = [23, \infty$, $L = [0, 13]$ and $I = (13, 23)$. A trajectory that starts in the state H, and makes an excursion into I is said to have a successful transition if it reaches state L before state H.

All trajectories are initialized in the state H at $x = 25$. The initial molecule production queue is assumed to be empty. A transient is computed for $\left(\tau^2 + 1\right) \times 10^4$ units of time. Any data is gathered after the transient. The mean residence times R_τ are computed by averaging over 10^4 transitions.

4.2.1 Stability analysis for the Positive Feedback Model

We analyze the spectrum of the linearization around the fixed point of the delay differential equation (4.6) rewritten as

$$\dot{x}(t) = B(x(t-\tau)) - D(x). \tag{4.7}$$

This equation is the deterministic counterpart of the stochastic positive feedback model examined in the manuscript. Linearizing Eq. (4.7) in the neighborhoods of a stable fixed point x_0 to yield a DDE

$$\dot{x}(t) = B'(x_0)(x(t-\tau) - x_0) + D'(x_0)(x(t) - x_0).$$

On setting $y(t) = x(t) - x_0$, $p = -B'(x_0)$, $q = D'(x_0)$, and assuming a solution of the form

$$y(t) = Ce^{st}$$

we get a characteristic equation

$$(s+q)e^{s\tau} + p = 0.$$

The k^{th}pair of eigenvalues s_k can be obtained by solving the equation

$$s_k = \frac{1}{\tau} W_k(-p\tau e^{\tau q}) - q$$

where W_k is the k^{th} branch of the Lambert W function. If s_0 is found to have negative real part, no other eigenvalues need to be computed as the stability is determined completely by s_0.

As is apparent from Fig. 4.5, as τ increases, the stability of both stable fixed points decreases. Therefore, while the stochastic positive feedback system becomes mores stable, its deterministic counterpart becomes less stable with an increase in delay.

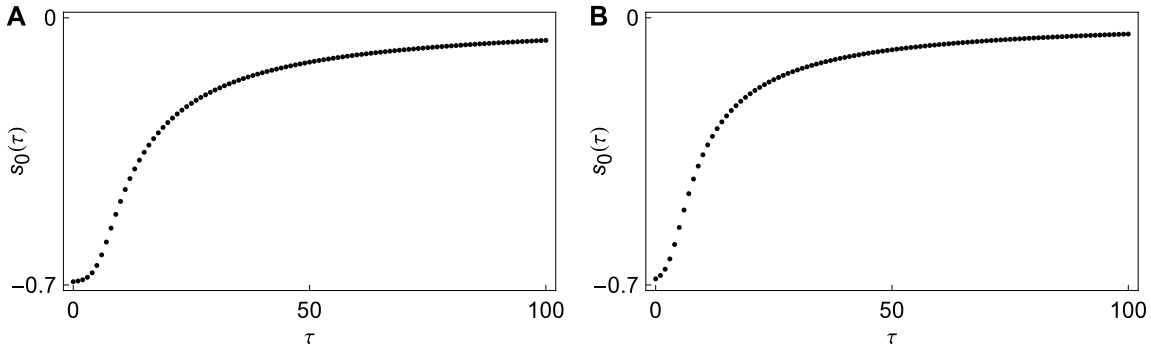

Figure 4.5: **Left:** Plot of the leading eigenvalue, as a function of the delay, for the linearization of the system in the neighborhood of the lower fixed point $x_0 = 7.21$. **Right:** Plot of the leading eigenvalue, as a function of the delay, for the linearization of the system in the neighborhood of the higher fixed point $x_0 = 36.01$

4.2.2 Distributed Delays

To examine the effect of distributed delay, we used Gamma delay distributions, $\kappa(\tau; \mu, \sigma)$, with different means and variances. The effect of increasing the mean of the distribution κ was qualitatively similar to increasing the magnitude of a constant delay, τ: The mean residence times increased sharply with small increases in this mean, and eventually appeared to saturate.

Increasing the variance of the gamma distribution for a fixed mean appears to initially slow down the rate of increase of the mean transition time with increasing delay; however, larger variances also appear to correspond to larger saturation values (see Fig. 4.6).

4.2.3 Delayed Deaths

We also consider a process that is formally constructed by delaying deaths (see Fig. 4.7). In such a model, as in the stochastic simulation algorithm with delayed births, the waiting

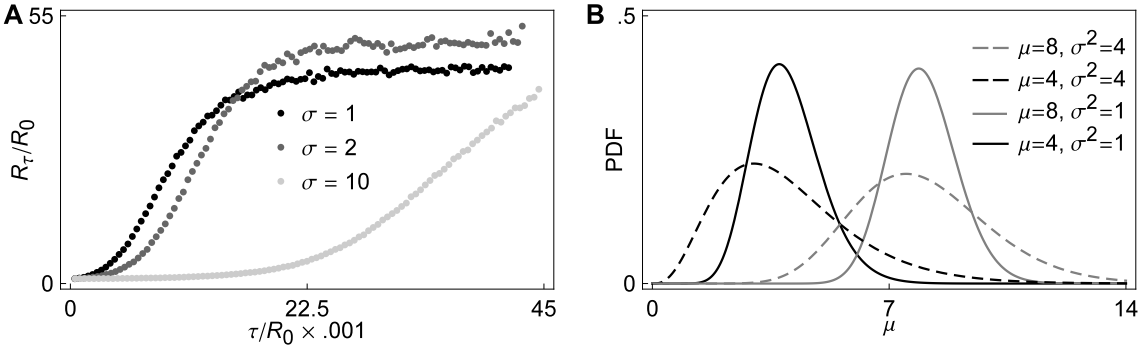

Figure 4.6: **Left: Positive feedback model with distributed delay.** A Gamma distribution with $\tau \in (0, 10)$ and $\sigma \in \{1, 2, 10\}$ is used for the positive feedback model. The parameters used are $a = 20, b = 5, k = 19$. The trajectories are initialized with a population of size 25, followed by a long transient. For small values of τ, the ratio R_τ / R_0 grows more slowly for larger variances. **Right: Delay distributions.** We use families of Gamma distributions with varying means and variances $\sigma^2 = 1, 4$. Displayed here are the distributions for means 4 and 8.

time to the next reaction is sampled. If that reaction is a birth type reaction, the population is updated immediately. For a degradation, we put the corresponding state change vector in a queue with a designated time of exit. A new reaction time is then sampled. Otherwise the algorithm is as described above.

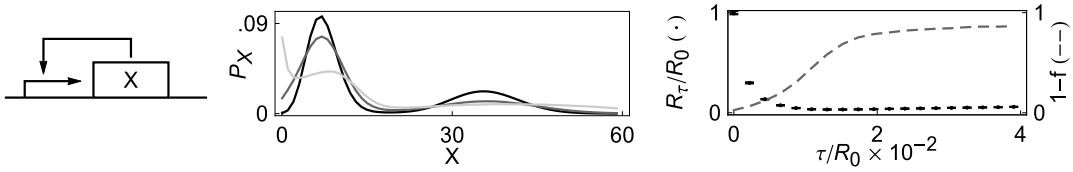

Figure 4.7: **Effect of delaying degradation.** *(Left)* The motif for the positive feedback model. *(Center)* The stationary distributions for three different delays. Darker lines correspond to larger delays. The maximum accumulation of proteins of type X increases with delay. Since the exiting state change vectors depend on the protein numbers at a time in the past, there is an accumulation of probability at $X = 0$ with increasing delay. *(Right)* Solid dots represent the mean residence times. Dashed lines represent the probability of succeeding in an attempted transition.

Delaying reactions that decrease population is less realistic, and can lead to negative population sizes. We disregard any reactions that would decrease the size of a population below 0. Formally, the deterministic equations approximating the stochastic dynamics in the case of delayed deaths can be written as

$$\dot{x} = \alpha + \beta \frac{x(t)^b}{c^b + x(t)^b} - \gamma x(t-\tau)$$

with the added constraint that $x(t) \geq 0$ for all $t \geq 0$.

Delaying deaths destabilizes the bistable switch. The explanation parallels that given in the main manuscript:Consider a large downward fluctuations from away from the upper stable state H. In the presence of transcriptional delay, birth rates are determined by the state $x(t-\tau)$, and the larger τ, the more likely it is that $x(t-\tau)$ is in state H. But death rates in state H are high and favor motion away from H. Therefore, the trajectory is pushed away from the stable state it came from. The result is a large decrease in residence times with increasing delay.

The RM described in the manuscript can capture this effect by appropriately changing the transition rates $\lambda^H_{I\to H}$ etc.

4.3 The lysis/lysogeny switch of phage λ

The lysis/lysogeny switch of phage-λ is realized as a set of chemical reactions involving multimerized forms of two transcription factors A, and B, which regulate gene expression by binding to the genome at an operator site O. The state of the operator is denoted by O if neither multimerized transcription factors are bound to it; when the multimer A_n is bound, the operator is denoted OA_n and when B_m is bound, the operator is denoted by OB_m.

A simplified model of the system consists of the following chemical reactions. The first two reactions represent the multimerization reactions of the transcription factors A and B. Multimerization is introduced into the model because transcription factors must bind to the DNA cooperatively to make a working switch.

$$nA \underset{k_b}{\overset{k_f}{\rightleftharpoons}} A_n$$

$$mB \underset{k_b}{\overset{k_f}{\rightleftharpoons}} B_m$$

The next two reactions represent the degradation of the transcription factor monomers as a first order reaction.

$$A \xrightarrow{\mu_A} \emptyset$$

$$B \xrightarrow{\mu_B} \emptyset$$

When no multimers are bound to the operator O, either A or B can be produced. This production is a result of a large number of biochemical steps, and for this reason, this reaction is assumed to be delayed. This delay is represented in the equations by setting τ as a superscript in the reaction rate constant. The synthesis reactions for A and B are then represented as

$$O \xrightarrow{k_A^{(\tau)}} O + A$$

$$O \xrightarrow{k_B^{(\tau)}} O + B.$$

The multimers A_n or B_m can bind reversibly to the operator O. This gives us

$$O + A_n k_{on} k_{off} O A_n$$

$$O + B_m k_{on} k_{off} O B_m.$$

Once the transcription factors of a certain kind bind to the operator, they allow only for the production of monomers of their own kind. This is represented by

$$OA_n \xrightarrow{k_A^{(\tau)}} OA_n + A$$

$$OB_m \xrightarrow{k_B^{(\tau)}} OB_m + B.$$

Together, this gives us a complete description of the lysis/lysogeny switch of phage-λ. We used the parameters $k_b = 5 = k_f = k_{on}, k_{off} = 1, k_A = 1 = k_B, \mu_A = 0.3 = \mu_B$ and $n = m = 2$. We assume the presence of only 1 operator O. A detailed analysis of the system, as well as a discussion of the model reduction of the deterministic approximation to a system of two ordinary differential equations can be found in [95].

The phase space of the lysis/lysogeny switch is mapped onto the RM as follows. We denote by N_X the number of molecules of type X in the system, we compute $|A| = N_A + 2N_{A_2}$ and $|B| = N_B + 2N_{B_2}$. If $|A| - |B| \geq 5$, the system is considered to be in state H, and if $|A| - |B| \leq 5$, in state L. Otherwise, if $|A| - |B| \in (-5, 5)$, the system is said to be in state I. The system is initialized with 5 molecules of the protein A and 30 molecules of A_2 (making $|A| = 65$) and $|B| = 0$. The initial molecule production queue is assumed to be empty. A transient is computed for $\left(\tau^2 + 1\right) \times 10^4$ units of time. Any data is gathered after the transient. Mean residence times are computed by averaging over 10^4 transitions.

4.4 Co-repressive toggle switch

We first describe the deterministic system in order to obtain the critical points. An appropriate choice of parameters is important, since transitions are very rare between stable points which are widely separated. We present here a geometrical method to easily find

feasible parameters.

The standard form of the co-repressive toggle switch with delayed production is given by the set of delay ordinary differential equations [26]

$$\dot{x} = \frac{\beta}{1 + y\left(t-\tau\right)^2/k} - \gamma x$$
$$\dot{y} = \frac{\beta}{1 + x\left(t-\tau\right)^2/k} - \gamma y.$$

The production and degradation propensity functions for the delayed Gillespie algorithm are obtained from these ODEs. Since the fixed points of the system do not change if delay is introduced, we assume that $\tau = 0$, and parametrize the critical points of the non-delayed system: Setting $\dot{x} = \dot{y} = 0$ and writing $a = \beta/2\gamma$ we get

$$0 = \frac{\beta}{1 + y^2/k} - \gamma x = \frac{2ak}{k + y^2} - x$$
$$0 = \frac{\beta}{1 + x^2/k} - \gamma y = \frac{2ak}{k + x^2} - y.$$

We now eliminate y between the two equations to obtain

$$\frac{\left(2ak - kx - x^3\right)\left(k + x^2 - 2xa\right)}{\left(k + x^2\right)^2 + 4ka^2} = 0.$$

The numerator is a polynomial of degree 5, which implies that there exist at most 5 real critical points. Solving the quadratic part of the above equations, and substituting back into the original equation leads to two critical points if $a > \sqrt{k}$:

$$\left(a - \sqrt{a^2 - k}, a + \sqrt{a^2 - k}\right)$$
$$\left(a + \sqrt{a^2 - k}, a - \sqrt{a^2 - k}\right).$$

On solving the cubic term explicitly, we observe that we get only one real solution. We arrange for this solution to be on the line $y = x$ by choosing $k = s^3/\left(2a - s\right)$ (in which

case the third critical point is (s, s)). On eliminating a, k from the above equations, we see that all critical points must lie on

$$xy\,(x + y - s) = s^3.$$

A stability analysis shows that for any s, the critical point at (s, s) is unstable, while the critical points $\left(a - \sqrt{a^2 - k}, a + \sqrt{a^2 - k}\right)$ and $\left(a + \sqrt{a^2 - k}, a - \sqrt{a^2 - k}\right)$ are stable.

Parameters can now be chosen by first choosing $s > 0$, then $a > s$ and finally $k = s^3 / (2a - s)$. We fix the parameter $\gamma = \ln(2)$ because we assume our units of time to be in terms of the protein half-life. Finally, we can solve for the parameter $\beta = 2\gamma a$.

The parameters used in our study are $s = 10, \gamma = \ln(2)$ and $a = 15.8202$. The phase space of the co-repressive toggle is mapped onto the RM as follows: Denote by X and Y the number of molecules of each protein type in the system. The region between the $y-$ axis and the 45° line that passes half-way between the saddle and the stable point in the region $Y > X$ is mapped onto the state H. The corresponding region between the x-axis and the 45° degree line in the $X > Y$ region is mapped into state L. All trajectories are initialized at the saddle (s, s). The initial molecule production queue is assumed to be empty. A transient is computed for $\left(\tau^2 + 1\right) \times 10^4$ units of time. Any data is gathered after the transient. All numerical estimates for mean residence times, failed transition probabilities, and stationary distributions are computed for 10^4 transitions from state H to L (and back).

Transition trajectories. Delay also widens the distribution of paths that lead to failed transitions, as well as the distribution of those paths that correspond to successful transitions. The changes in the densities of the failed transition paths appear to be more sensitive to delay. Since the RM is not constructed using the specific features of any of the

models, this effect cannot be explained using our reduction. We present in Figure 4.8 the densities of the paths that correspond to failed, and successful transitions, for the co-repressive toggle switch.

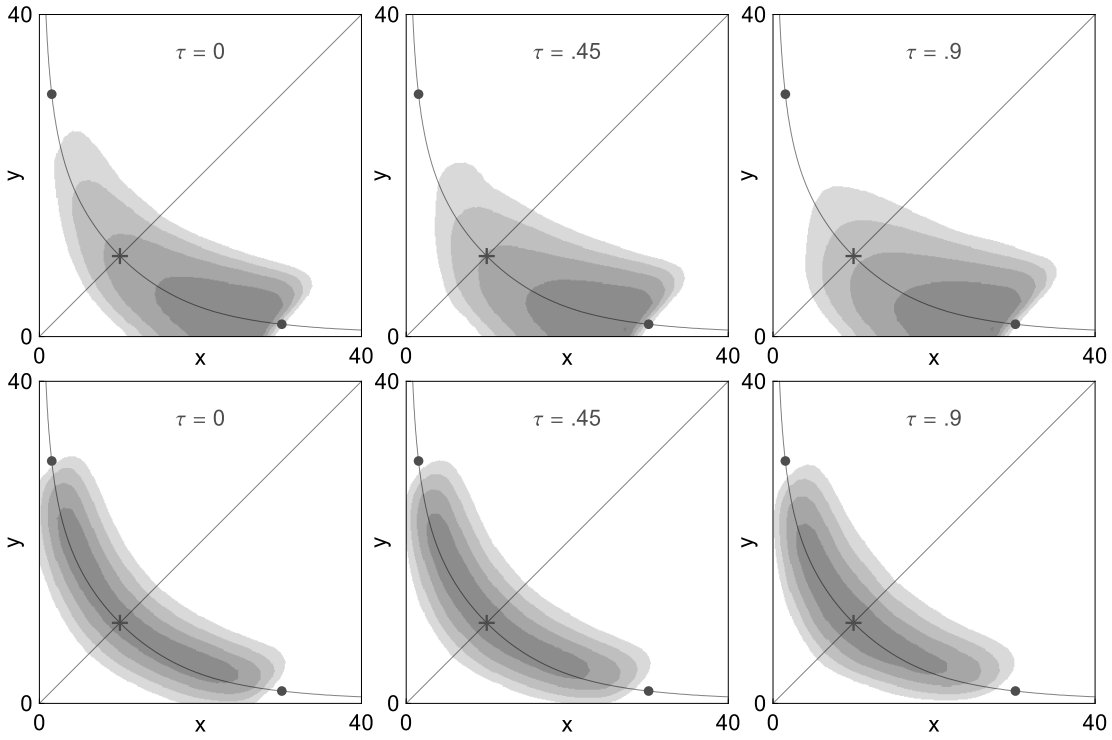

Figure 4.8: The top panels show the density corresponding to trajectories that make a failed transition attempt, starting in a neighborhood of the stable point in the region $X > Y$. With increasing τ, the support of the density is more spread out. The bottom panels show the densities for the trajectories corresponding to successful transitions. Again, we observe that the support of the densities widens, although the effect is not as pronounced as in the case of failed transitions.

4.5 Discussion

The existence of multiple stable states is a common feature of genetic networks, such as those that determine cell fate in multicellular organisms [42]. Since noise is ubiquitous in gene networks, mechanisms that stabilize dynamics are essential. We have shown that transcriptional delay can stabilize bistable gene networks. The tendency to return to the state from which the system just escaped increases with delay.

The RM proposed in this chapter depends on neither the explicit underlying model, the specific distribution of delay times, nor on the exact mechanism that produces the delay. Therefore, we conclude: (a) That distributed delay will stabilize bistable systems, provided the delay distribution is not close to the residence times of the system; as shown in Fig. 4.6, this is indeed the case; (b) A *decrease* in stability can be observed by delaying degradation and keeping production instantaneous (Fig. 4.7). The explanation given in Fig. 4.2 carries over to this case: Delayed death pushes the system away from the stable state that was just visited; and (c) The particular details of the mechanism that produce the delay are not important (these mechanisms are yet to be characterized completely experimentally). Hence, the effect should be observable even in more detailed models that explicitly model sequential protein production.

In some previous studies of bistable systems in the presence of delay, it was assumed that the delay time was on the order of the residence time [60, 89]. In this case, it was sufficient to consider reductions with only two states, *e.g.*, H and L. However, transcriptional delays in gene networks are typically much smaller than residence times. As we have shown here, in such systems it is therefore appropriate to introduce an additional intermediate state, *e.g.* I, in a reduced model.

Transcriptional delay introduces a number of other dynamical changes to stochastic bistable systems. For instance, typical transition paths between the stable states change with delay (Fig. 4.8). Such changes depend on the specifics of the individual models, and cannot be understood using the reduction described above. Precise results for large deviations in delayed Langevin equations have been obtained [24, 74]. However, the correct Langevin approximations of the models we consider here need to include delay in the diffusion term [12, 83]). Extensions of previous large deviation approaches may provide more detailed information about transitions between states in particular models of genetic switches.

Chapter 5

Delayed Stochastic Differential Equations

In this Chapter[1], we establish a link between delayed birth-death (dBD) processes and delayed langevin equations (dCLEs) by showing that the distance between the dBD process and the correct approximating dCLE process converges to zero as system size tends to infinity (as measured by expectations of functionals of the processes). In particular, this result applies to all moments. It is natural to express distance in terms of expectations of functionals because the dBD process is spatially discrete while the correct dCLE produces continuous trajectories (see Figure 5.1). This first result is just a corollary of our main mathematical theorem. We estimate the probability that a trajectory of the correct dCLE remains inside a narrow tube around the corresponding trajectory of the dBD process. This quantitative tube estimate will allow practitioners to determine the accuracy of

[1]Content in this Chapter has been previously published: Gupta, C., López, J. M., Azencott, R., Bennett, M. R., Josić, K., Ott, W. (2014). **Modeling delay in genetic networks: From delay birth-death processes to delay stochastic differential equations**. *The Journal of Chemical Physics*, 140(20), 204108.

simulations of concrete biochemical reaction networks. The mathematical results hold for both fixed delay and distributed delay (see Figure 5.2A).

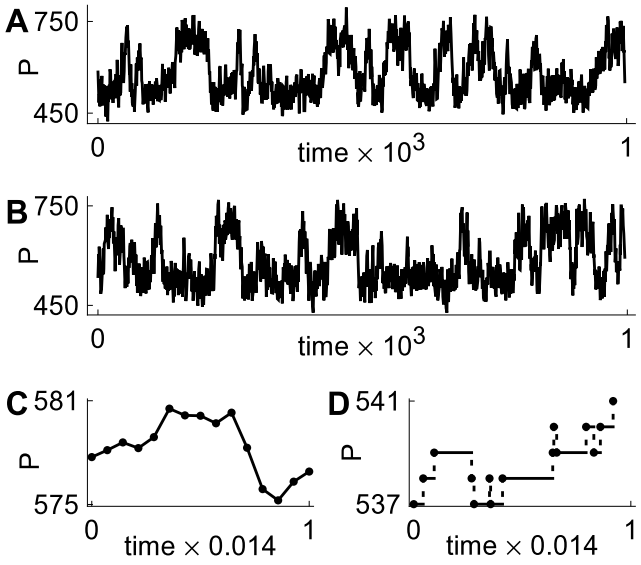

Figure 5.1: **(A)** Typical trace obtained by using the dCLE. **(B)** Corresponding trace obtained using dSSA. By zooming into a small time segment, we see that the trace obtained from the dCLE **(C)** consists of equi-spaced time points (corresponding to an Euler discretization) and is continuous. The corresponding segment of the trace from the dSSA (D) consists of events that occur at random times and has jump discontinuities. The displayed traces were gathered after a long transient. The model simulated is the single-gene positive feedback model with $N = 500$; see Section 5.1.3.1.

The correct dCLE approximation is distinguished within the class of Gaussian approximations of the dBD process by the fact that it matches both the first and second moments of the dBD process. As we will see, it performs remarkably well at moderate system sizes in a number of dynamical settings: steady state dynamics, oscillatory dynamics, and certain metastable switches. We will demonstrate via simulation and argue mathematically using characteristic functions that no other Gaussian process with appropriately scaled noise performs as well. In the following, the term 'dCLE' shall refer specifically to the

dCLE derived by Brett and Galla and expressed by (5.15), unless specifically stated otherwise.

5.1 Simulations and interpretation of results

Genetic regulatory networks may be simulated using an exact dSSA to account for transcriptional delay as we have seen in previous chapters. Here we provide a heuristic derivation of a related dCLE, and show that in a number of concrete examples it provides an excellent approximation of the system (see Figure 5.2). These simulations raise the following questions: Is the dCLE approximation valid in general? Can the expected quality of the approximation be quantified in general? We answer these questions mathematically in Section 5.2.

We will adopt the following notation for reactions with delay,

$$X + Y \overset{\alpha(X,Y)}{\underset{\mu}{\dashrightarrow}} Z$$

Here $\alpha(X, Y)$ denotes the rate of the reaction, the dashed arrow indicates a reaction with delay, and μ is a probability measure that describes the delay distribution. Solid arrows indicate reactions without delay.

5.1.1 A transcriptional cascade

First we consider a transcriptional cascade with two genes that code for proteins X and Y. Protein X is produced at a basal rate; production of Y is induced by the presence of X. The state of the system is represented by an ordered pair (X, Y). Note that we use X and Y to denote both protein names and protein numbers. The reactions in the network, and

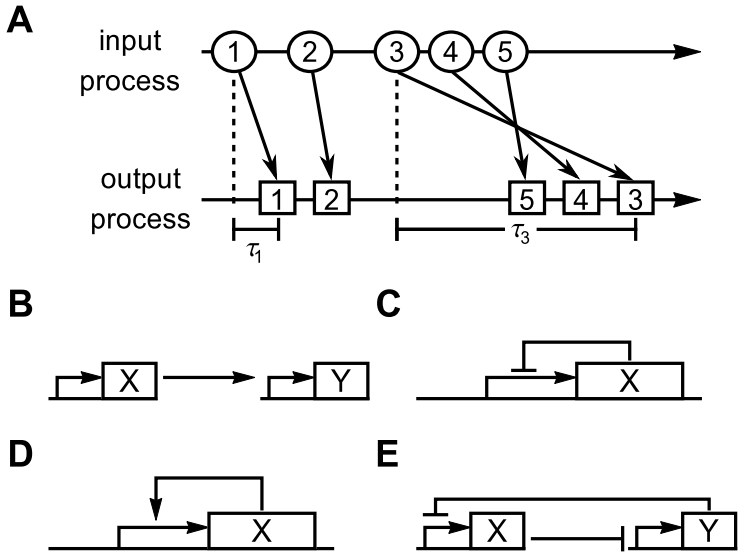

Figure 5.2: **(A)** The effect of distributed delay on the protein production process. The "input process" is the first step in the transcription process, while the "output process" is the final mature product that enters the population. The time delay τ accounts for the lag between the initialization of transcription and the production of mature product. In a system with distributed delay, different production events can have different delay times; the order of the output process may therefore not match that of the input process. **(B–E)** Simulated gene regulatory network motifs: a transcriptional cascade **(B)**, oscillators **(C)** and metastable systems **(D–E)**.

the associated state change vectors v_i, are given by

$$\emptyset \xrightarrow[\mu]{a} X \qquad v_1 = (1, 0) \tag{5.1}$$

$$X \xrightarrow{b_1 X} \emptyset \qquad v_2 = (-1, 0) \tag{5.2}$$

$$\emptyset \xrightarrow[\mu]{\psi(X)} Y \qquad v_3 = (0, 1) \tag{5.3}$$

$$Y \xrightarrow{b_2 Y} \emptyset \qquad v_4 = (0, -1) \tag{5.4}$$

This system can be simulated exactly using the dSSA: Suppose the state of the system and the reactions in the queue are known at time t_0 (the queued reactions can be thought of as

the "input process"; see Figure 5.2A), and that the delay kernel μ is supported on a finite interval $[0, \tau_0]$.

1. Sample a waiting time t_w from an exponential distribution with parameter $A_0 a + b_1 X + \psi(X) + b_2 Y$.

2. If there is a reaction in the queue that is scheduled to exit at time $t_q < t_0 + t_w$, advance to time t_q and set $t_0 \to t_q$ and $(X, Y) \mapsto (X, Y) + \left(v_i^{(1)}, v_i^{(2)}\right)$, where $v_i = \left(v_i^{(1)}, v_i^{(2)}\right)$ is the change in the system due to the scheduled reaction. Finally, sample a new waiting time for the next reaction.

3. If no reaction exits before $t_0 + t_w$, set $t_0 \mapsto t_0 + t_w$ and sample a reaction type from the set $\{1, 2, 3, 4\}$ with probabilities proportional to $\{a, b_1 X, \psi(X), b_2 Y\}$, respectively. If the reaction chosen is a non-delayed reaction, perform the update $(X, Y) \mapsto (X - 1, Y)$ (or $(X, Y) \mapsto (X, Y - 1)$). However, if the reaction chosen is a delayed reaction, the state change vector $(1, 0)$ (or $(0, 1)$) is put into the queue along with an exit time t_q. The difference $\tau = t_q - t_0$ between the current and the exit time is sampled from the delay distribution μ.

We now heuristically derive the dCLE for the feed-forward system from this spatially discrete process.

Suppose the delay kernel μ is given by a probability density function κ supported on $[0, \tau_0]$ ($d\mu(s) = \kappa(s)\, ds$). We first approximate the number of reactions that produce Y (Eq. (5.3)) that will be completed within the interval $[t_0, t_0 + \Delta]$, where t_0 denotes the current time and Δ is a small increment. Since the production of Y involves delay, a reaction of this type that is completed within $[t_0, t_0 + \Delta]$ must have been initiated at some time within $[t_0 - \tau_0, t_0]$. Let $t_0 > t_0 - \Delta > t_0 - 2\Delta > \cdots$ be the partition of $[t_0 - \tau_0, t_0]$ into intervals

of length Δ. The (random) number of reactions completed within $[t_0, t_0 + \Delta]$ *and* initiated within $[t_0 - (i+1)\,\Delta, t_0 - i\Delta]$ may be approximated by a Poisson random variable with mean

$$\psi\left(X_{t_0-(i+1)\Delta}\right)\Delta \cdot \kappa\left(\left(i+1\right)\Delta\right)\Delta.$$

Summing over i, the (random) number of reactions completed within $[t_0, t_0 + \Delta]$ may be approximated by a Poisson random variable with mean

$$\Delta \sum_i \left[\psi\left(X_{t_0-(i+1)\Delta}\right)\left(\kappa\left(\left(i+1\right)\Delta\right)\Delta\right)\right];$$

this is a Riemann sum that approximates the integral

$$\Delta \int_0^{\tau_0} \psi\left(X_{t_0-s}\right)\kappa\left(s\right)\, ds.$$

Known as τ-leaping, this line of reasoning produces a Poissonian approximation of the dBD process:

$$\begin{aligned}\delta X_t &= \text{Poisn}\left(a\Delta\right) - \text{Poisn}\left(b_1 X_t \Delta\right)\\ \delta Y_t &= \text{Poisn}\left(\Delta \int_0^{\tau_0} \psi\left(X_{t-s}\right)\, d\mu\left(s\right)\right) - \text{Poisn}\left(b_2 Y_t \Delta\right).\end{aligned}$$

Here $\text{Poisn}\left(\eta\right)$ denotes a Poisson random variable with mean η.

If these Poisson random variables have large mean, they can be approximated by normal random variables. For example, the Poisson variable representing the number of reactions that produce Y can be approximated by a normal random variable with mean and variance equal to $\Delta \int_0^{\tau_0} \psi\left(X_{t-s}\right)\, d\mu\left(s\right)$. Since each reaction changes the state of either X or Y (but never both), it follows that the evolution of the system can be approximated

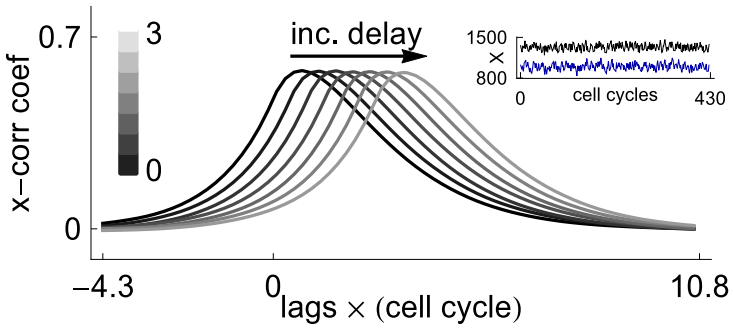

Figure 5.3: **Cross correlation functions for the two-species feed-forward architecture at system size** $N = 1000$. The inset shows sample trajectories for X (black; top) and Y (blue; bottom). The mean field model has a fixed point, and for this reason, stochastic dynamics stay within a neighborhood of this fixed point (see Theorem 1). However, the effect of increasing delay is clearly seen in the cross correlation function. The color bar displays the delay size corresponding to the cross correlation curves. Parameter values are given by $\tilde{\psi}(X) = \tilde{a}_1 x^3 / \left(m^3 + x^3\right)$, $\tilde{a} = 0.92$, $\tilde{a}_1 = 1.39$, $b_1 = b_2 = \ln(2)$, $m = 1.33$, $\mu = \delta_\tau$. The cross correlations have been normalized by dividing by the standard deviations σ_X and σ_Y of X and Y. Time has been normalized to cell cycle length.

by the stochastic difference equation

$$\delta X_t = \Delta\left(a - b_1 X_t\right) + \sqrt{\Delta\left(a + b_1 X_t\right)} N(0,1) \tag{5.5a}$$

$$\delta Y_t = \Delta\left(\int_0^{\tau_0} \psi\left(X_{t-s}\right)\, d\mu(s) - b_2 Y_t\right) \tag{5.5b}$$

$$+ \sqrt{\Delta\left(\int_0^{\tau_0} \psi\left(X_{t-s}\right)\, d\mu(s) + b_2 Y_t\right)} N(0,1), \tag{5.5c}$$

where $N(0,1)$ is the standard normal random variable.

System (5.5) may be written in terms of concentrations. Let N be a system size parameter. We think of N as a characteristic system size; X_t/N and Y_t/N therefore represent fractions of this characteristic value. Writing $x_t = X_t/N$, $y_t = Y_t/N$, $\tilde{\psi}(x) = \psi(Nx)/N$,

and assuming that the basal production rate a scales with N as $a = \tilde{a}N$, we obtain

$$\delta x_t = \Delta\left(\tilde{a} - b_1 x_t\right) + \frac{1}{\sqrt{N}}\sqrt{\Delta\left(\tilde{a} + b_1 x_t\right)}N\left(0,1\right) \tag{5.6a}$$

$$\delta y_t = \Delta\left(\int_0^{\tau_0} \tilde{\psi}\left(x_{t-s}\right)\, d\mu\left(s\right) - b_2 y_t\right) \tag{5.6b}$$

$$+ \frac{1}{\sqrt{N}}\sqrt{\Delta\left(\int_0^{\tau_0} \tilde{\psi}\left(x_{t-s}\right)\, d\mu\left(s\right) + b_2 y_t\right)}N\left(0,1\right). \tag{5.6c}$$

Eq. (5.6) is the Euler–Maruyama type discretization of a delay stochastic differential equation. Replacing Δ with dt and $\sqrt{\Delta}N\left(0,1\right)$ with dW_t in (5.6), we obtain

$$dx_t = \left(\tilde{a} - b_1 x_t\right)\, dt + \frac{1}{\sqrt{N}}\sqrt{\left(\tilde{a} + b_1 x_t\right)}\, dW_t^1 \tag{5.7a}$$

$$dy_t = \left(\int_0^{\tau_0} \tilde{\psi}\left(x_{t-s}\right)\, d\mu\left(s\right) - b_2 y_t\right) dt \tag{5.7b}$$

$$+ \frac{1}{\sqrt{N}}\sqrt{\left(\int_0^{\tau_0} \tilde{\psi}\left(x_{t-s}\right)\, d\mu\left(s\right) + b_2 y_t\right)}\, dW_t^2. \tag{5.7c}$$

This is the dCLE for the transcriptional cascade in this section.

Taking the formal thermodynamic limit, $N \to \infty$, in Eq. (5.7) yields the reaction rate equations derived in [73]:

$$\mathrm{d}x_t = \left(\tilde{a} - b_1 x_t\right)\mathrm{d}t \tag{5.8a}$$

$$\mathrm{d}y_t = \left(\int_0^{\tau_0} \tilde{\psi}\left(x_{t-s}\right)\, \mathrm{d}\mu\left(s\right) - b_2 y_t\right)\mathrm{d}t. \tag{5.8b}$$

The dynamics described by Eq. (5.8) are quite simple; if $b_1 > 0$ and $b_2 > 0$, then (5.8) has a globally attracting stable stationary point.

To test the validity of the dCLE approximation (5.7), we examine if it captures the interaction between the two proteins in our transcriptional cascade network. Figure 5.3 shows the cross correlation functions obtained by simulating the system with $N = 1000$ using dSSA. From left to right, the curves correspond to fixed delay τ increasing from

0 to 3. The corresponding cross correlation curves for the dCLE approximation (5.7) are indistinguishable from those obtained using dSSA.

In the heuristic derivation above, we first fix N and let $\Delta \to 0$ to obtain the dCLE; we then separately let $N \to \infty$ to obtain the thermodynamic limit. Brett and Galla [12] also derive the dCLE by first fixing N and then sending $\Delta \to 0$. However, the two limits, $\Delta \to 0$ and $N \to \infty$, cannot be taken independently; this is a common problem with heuristic derivations of stochastic differential equations, even in the absence of delay [23]. The time discretization, Δ, can be thought of as a sampling frequency, while the system size, N, determines the rate at which reactions fire. If N becomes too large for a given Δ, then the number of reactions that fire within $[t, t + \Delta]$ no longer follows a Poisson distribution with mean dependent only on the state of the system at time t. On the other hand, if N is too small for a given Δ, then the Poisson distribution cannot be approximated by a normal distribution. In order to rigorously derive the Langevin approximation and estimate the distance between the dBD and dCLE processes, we will have to take a careful limit by relating Δ to N (with $\Delta \to 0$ as $N \to \infty$). We describe the proper scaling in Section 5.2.

Applied to the transcriptional cascade, Theorem 1 in 5.2 provides explicit bounds for the probabilities that the dBD and dCLE processes deviate from a narrow tube around the solution of Eq. (5.8).

In the previous example, the deterministic system has a fixed point. The time series for the stochastic system therefore stay within a small neighborhood of this fixed point (see inset, Fig. 5.3). In the next example, we show that the dCLE approximation remains excellent even when the deterministic dynamics are non-trivial. We consider a degrade-and-fire oscillator for which the deterministic system has a limit cycle. The dCLE correctly

captures the peak height and the inter-peak times for the dSSA realization of the degrade and fire oscillator, in addition to statistics such as the mean and variance. The approximation does not break down at small instantaneous protein numbers. Indeed, the mathematical theory developed in this work makes an important point: protein concentrations at any particular time do not limit the quality of the dCLE approximation (in the presence of delay, or otherwise). Instead, the quality of the dCLE approximation depends on the latent parameter N. Theorem 1 makes this more precise: if one fixes the allowable error ε in the approximation of the dBD process by the dCLE process, then the time T during which the approximation error stays smaller than ε increases with N.

5.1.2 Degrade and fire oscillator

The degrade and fire oscillator depicted schematically in Figure 5.2C consists of a single autorepressive gene and corresponds to the reaction network

$$\emptyset \overset{\psi(X)}{\underset{\mu}{\dashrightarrow}} X \xrightarrow{\gamma X} \emptyset$$

$$X + E \xrightarrow{\eta(X)} E$$

The production rate $\psi(X)$ is given by $\psi(X) = Nf(X/N)$, where f is the propensity function

$$f(x) = \frac{\alpha}{1 + (x/C_1)^4};$$

the enzymatic degradation rate $\eta(X)$ is given by $\eta(X) = Ng(X/N)$. Here $g(x) = V_{max}x/(K + x)$; K is the Michaelis-Menten constant, V_{max} the maximal enzymatic degradation rate, and γ the dilution rate coefficient. In the thermodynamic limit, the system is modeled by the delay differential equation

$$\frac{dx}{dt} = \int_0^{\tau_0} \frac{\alpha}{1 + \left(\frac{x(t-s)}{C_1}\right)^4} d\mu(s) - \gamma x(t) - \frac{V_{max}x(t)}{K + x(t)}. \tag{5.9}$$

As before, $x(t)$ denotes the concentration of protein X. We model the formation of functional repressor protein using distributed delay (described by the probability measure μ); this delayed negative feedback can induce oscillations [63]. Figure 5.4A depicts a sample realization of the stochastic version of the degrade and fire oscillator (the finite system size regime) generated by dSSA.

The dCLE approximation is in this case given by

$$\begin{aligned} dx_t = &\int_0^{\tau_0} f(x_{t-s})\, d\mu(s) - \gamma x_t - g(x_t)\, dt \\ &+ \frac{1}{\sqrt{N}} \left[\int_0^{\tau_0} f(x_{t-s})\, d\mu(s) + \gamma x_t + g(x_t) \right]^{\frac{1}{2}} dW_t. \end{aligned} \tag{5.10}$$

Figure 5.4 illustrates that Eq. (5.10) provides a good approximation of the dBD dynamics, even when system size is relatively small. At system size $N = 50$, the spike height distribution and interspike interval distribution obtained using the dSSA (black dots in Figure 5.4B– 5.4C) are nearly indistinguishable from those obtained using Eq. (5.10) (black curves in Figure 5.4B–5.4C). Further, we see a close match with respect to mean repressor protein level and repressor protein variance across a range of system sizes (Figure 5.4D–5.4E).

Interestingly, the dCLE approximation is very good even though the protein number approaches zero during part of the oscillation. This illustrates a central feature of the theory: the quality of the dCLE approximation is a function of a latent parameter N, not of the number of molecules present at any given time.

The exact form of the diffusion term is crucial to the accuracy of the dCLE approximation. If we remove delay from the diffusion term in Eq. (5.10), we obtain

$$\begin{aligned} dx_t = &\int_0^{\tau_0} f(x_{t-s})\, d\mu(s) - \gamma x_t - g(x_t)\, dt \\ &+ \frac{1}{\sqrt{N}} \left[f(x_t) + \gamma x_t + g(x_t) \right]^{1/2} dW_t. \end{aligned} \tag{5.11}$$

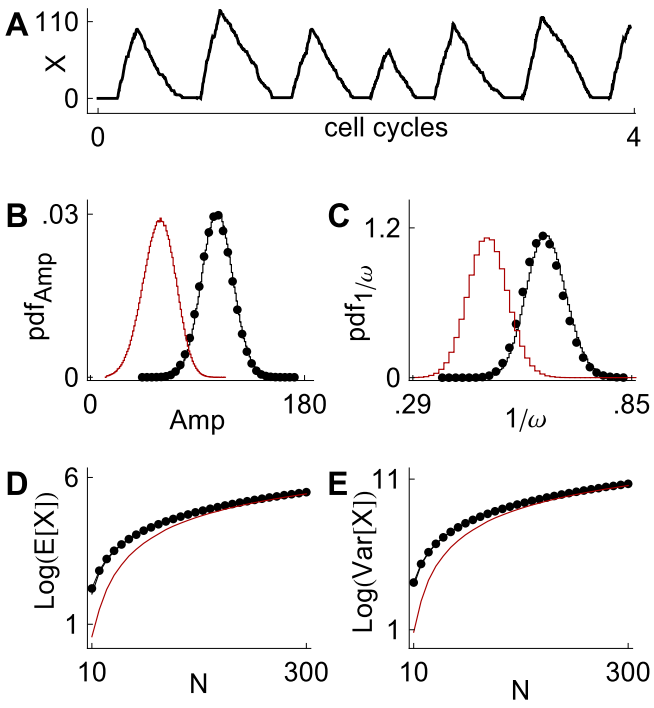

Figure 5.4: **Comparison of dSSA results to dSDE approximations for the degrade and fire oscillator.** (A) depicts a stochastic realization of the oscillator generated by dSSA. We compare dSSA statistics (black dots in (B)–(E)) to those generated by the dCLE approximation given by Eq. (5.10) (black curves). We also show results for the dSDE approximation given by Eq. (5.11), obtained by removing delay from the diffusion term in Eq. (5.10) (red curves). At system size $N = 50$, the dCLE approximation given by Eq. (5.10) closely matches dSSA with respect to spike height distribution (B) and interspike interval distribution (C). In contrast, removing delay from the diffusion term results in a poor approximation of these distributions, as shown by the sizable shifts affecting the red curves. (D) and (E) illustrate mean repressor protein level and repressor protein variance, respectively, as functions of system size N. Eq. (5.10) provides a good approximation for all simulated values of N while the performance of Eq. (5.11) improves as N increases. The quantity P represents protein number, not protein concentration. Parameter values are $\alpha = 20.8$, $C_1 = 0.04$, $\beta = \ln(2)$, $V_{max} = 5.55$, $\gamma_0 = 0.01$, $\mu = \delta_{0.14}$. A soft boundary was added at 0 to ensure positivity.

At system size $N = 50$ (red curves in Figure 5.4B–5.4C), dSDE (5.11) produces dramatically different results from those generated by the correct dCLE approximation. The

performance of Eq. (5.11) improves as N increases (Figure 5.4D–5.4E). This is expected, as both Eq. (5.10) and Eq. (5.11) converge weakly to Eq. (5.9) as $N \to \infty$.

5.1.3 Metastable systems

Here we examine two canonical examples to show that for these systems, the dCLE can be used to study the impact of delay on metastability: a positive feedback circuit and a co-repressive genetic toggle switch.

We offer a cautionary note before we proceed with the examples. Even in the Markovian context, SDE and Fokker-Planck modeling does not always accurately capture large deviations statistics associated with an underlying spatially-discrete stochastic process. Existing work for systems without delay (see *e.g.* [21, 38, 49]) illustrates the delicate relationship between continuum and discrete treatments of hitting times, extinction times, and the like. Determining when the dCLE accurately approximates the large deviations statistics of the underlying dBD process is an open problem.

5.1.3.1 Single species positive feedback circuit

The simplest metastable system consists of a single protein that drives its own production (Figure 5.2D). The chemical reaction network is given by

$$\emptyset \xrightarrow[\mu]{\psi(X)} X \xrightarrow{\gamma X} \emptyset$$

with $\psi(X) = Nf(X/N)$ for the propensity

$$f(x) = \alpha + \frac{\beta x^b}{c^b + x^b}.$$

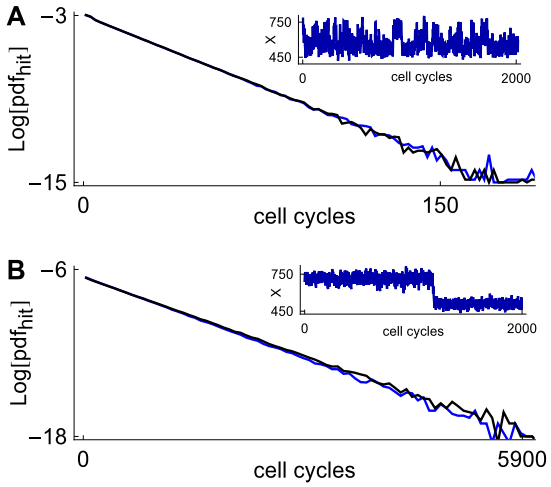

Figure 5.5: **Hitting time distributions for the positive feedback circuit**. Black curves represent dSSA data; blue curves represent data from the dCLE approximation. The top panel corresponds to the Markov case $\tau = 0$ and a Hill coefficient of $b = 15$. The bottom row corresponds to a delay $\tau = 0.75$ and $b = 25$. The tail of the hitting times distribution becomes longer with both increasing delay and increasing Hill coefficient b. The dCLE captures the lengthening of the tail due to both effects. Parameter values are $\alpha = 0.35$, $\beta = 0.15$, $c = 0.615$, $\gamma = \ln(2)$, $\mu = \delta_\tau$, $N = 1000$.

In the thermodynamic limit, the dynamics of this model are described by the DDE

$$\mathrm{d}x_t = \int_0^{\tau_0} \alpha + \beta \frac{x\left(t-s\right)^b}{c^b + x\left(t-s\right)^b} \,\mathrm{d}\mu\left(s\right) - \gamma x\left(t\right)\,\mathrm{d}t. \tag{5.12}$$

Here x represents protein concentration and b is the Hill coefficient. In the thermodynamic limit, there are two stable stationary states, x_l and x_h, as well as an unstable stationary state x_s. These states satisfy $x_l < x_s < x_h$.

In the stochastic (finite N) regime, the stationary states x_l and x_h become metastable. We simulate the metastable dynamics using dSSA and the dCLE approximation (5.15) given in this case by

$$\mathrm{d}x_t = \int_0^{\tau_0} f\left(x_{t-s}\right)\,\mathrm{d}\mu\left(s\right) - \gamma x_t\,\mathrm{d}t + \frac{1}{\sqrt{N}}\left[\int_0^{\tau_0} f\left(x_{t-s}\right)\,\mathrm{d}\mu\left(s\right) + \gamma x_t\right]^{\frac{1}{2}} \mathrm{d}W_t. \tag{5.13}$$

Figure 5.5 displays hitting time distributions for the dSSA simulations (black curves) and Eq. (5.13) (blue curves). A hitting time is defined as follows: We choose neighborhoods $(x_l - \delta_l, x_l + \delta_l)$ and $(x_s - \delta_s, x_s + \delta_s)$ of x_l and x_s, respectively. We start the clock when a trajectory enters $(x_l - \delta_l, x_l + \delta_l)$ from the right. The clock is stopped when that trajectory first enters $(x_s - \delta_s, x_s + \delta_s)$. A hitting time is the amount of time that elapses from clock start to clock stop.

We see that for no delay (Figure 5.5, top) and fixed delay $\tau = 1$ (bottom), the dCLE approximation accurately captures the hitting time distributions for Hill coefficients increasing from 15 to 25. Hence, the dCLE approximation accurately captures the rare events associated with a spatially-discrete delay stochastic process. This is significant because dSDEs are more amenable to large deviations theoretical analysis than their spatially-discrete counterparts.

Hitting times increase dramatically as the delay increases from 0 to 1, in accord with the analysis in the previous chapter. A dramatic increase is also seen as the Hill coefficient increases. This is due to the fact that the potential wells around x_l and x_h deepen as b increases.

5.1.3.2 Co-repressive toggle switch

The co-repressive toggle switch (Figure 5.2E) is a two-dimensional metastable system described in the thermodynamic limit by the DDEs

$$\mathrm{d}x_t = \int_0^{\tau_0} \frac{\beta}{1 + y\,(t-s)^2/k}\,\mathrm{d}\mu_1(s) - \gamma x\,\mathrm{d}t \tag{5.14a}$$

$$\mathrm{d}y_t = \int_0^{\tau_0} \frac{\beta}{1 + x\,(t-s)^2/k}\,\mathrm{d}\mu_1(s) - \gamma y\,\mathrm{d}t. \tag{5.14b}$$

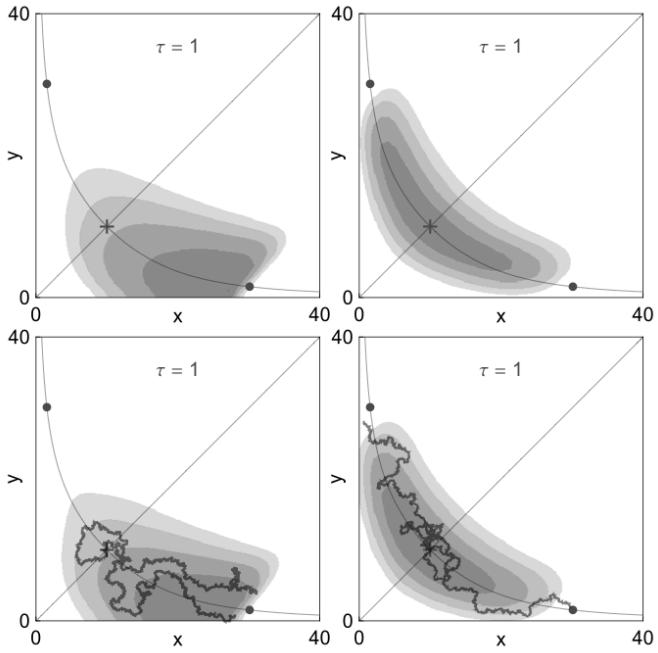

Figure 5.6: **Conditional density plots for the co-repressive toggle switch**. The two left panels illustrate trajectories that leave a neighborhood of the stable point (x_h, y_l) and fall back into the same neighborhood before transitioning into a neighborhood of the stable point (x_l, y_h); displayed densities are conditioned on this set of trajectories. The two right panels illustrate trajectories that leave a neighborhood of (x_h, y_l) and transition to a neighborhood of (x_l, y_h) before falling back into the first neighborhood; displayed densities are conditioned on the set of such trajectories. Top panels correspond to dSSA; bottom panels correspond to dCLE. Cartoons of typical trajectories corresponding to failed transitions (left) and successful transitions (right) are shown for the dCLE process. Plots are shown for $N = 30$. The values of the other parameters are $\beta = 0.73$, $k = 0.05$, $\gamma = \ln(2)$.

The measure μ_1 describes the delay associated with production in this symmetric circuit. Eq. (5.14) has two stable stationary points (x_l, y_h) and (x_h, y_l) separated by the unstable manifold associated with a saddle equilibrium point (x_s, y_s). In the stochastic (finite system size) regime, the stable stationary points become metastable. In this regime a typical trajectory spends most of its time near the metastable points, occasionally moving between them.

Figure 5.6 displays density plots corresponding to trajectories that either successfully transition between metastable states (four panels on the right) or make failed transition attempts (four panels on the left). Even for the moderate system size $N = 30$, the density plots generated by dSSA (top four panels) closely match those generated by the dCLE approximation in Eq. (5.15) (bottom four panels).

Given the importance of rare events throughout stochastic dynamics, it is encouraging that the dCLE approximation captures well the dynamics of two very popular biochemical motifs of metastability.

5.2 Main Mathematical Results

The simulations thus far described suggest that the dCLE closely approximates the dBD process provided that the time increment Δ scales properly with N. Here we mathematically quantify the quality of the approximation and we determine the optimal scaling of Δ.

The general biochemical setup is as follows. Consider a reaction network of D biochemical species and M possible reactions. We are interested in describing the dynamics as a function of a latent system parameter N, the system size. The state of the system at time t is described by the vector $B_N(t) \in Z^D$ that lists the number of molecules of each species present at time t.

Each reaction R_j is described by the following:

- A propensity function $f_j : R^D \to R^+$. The firing rate of reaction R_j is given by $Nf_j(B_N(t)/N)$. It is reasonable to use propensity functions to model firing rates in

well-mixed chemical systems [29]. We assume the propensities satisfy mild technical conditions [37]; these conditions are essentially always satisfied by chemical reaction networks. Examples include reactions of first and second order, Michaelis-Menten kinetics, and propensities of Hill type.

- A state-change vector $v_j \in Z^D$. The vector v_j describes the change in the number of molecules of each species that results from the completion of a reaction of type j.

- A probability measure μ_j that models the delay between the initiation and completion of a reaction of type j. We allow the delay to be distributed (μ_j has a density) or fixed (μ_j is a Dirac mass δ_τ at a fixed delay τ). For instantaneous reactions, $\mu_j = \delta_0$.

In order for the dCLE to closely approximate the dBD process, the time step Δ must be small enough so that the propensity functions remain essentially constant on time intervals of length Δ and large enough so that many reactions fire on such intervals. Gillespie [29] calls such a Δ a *macroscopic infinitesimal time* for the reaction network. The primary mathematical result of this paper provides a scaling of Δ as a function of N for which Δ is rigorously shown to be a macroscopic infinitesimal time for the reaction network. Further, we quantify the quality of the dCLE approximation: with high probability, the solution trajectory of the dCLE will remain inside a narrow tube around the solution trajectory of the dBD process. We now state the main result precisely and then discuss implications.

Since $B_N(t)$ is a vector of molecule numbers, we rescale by N to allow for a direct comparison of the dBD process and the dCLE. The correct dCLE approximation of $(b_N(t)) = (B_N(t)/N)$ is given for $1kD$ by

$$dx_k = \left(\sum_{j=1}^{M} \int_0^{\tau_0} v_{jk} f_j\left(x\left(t-s\right)\right) \, d\mu_j\left(s\right) \right) dt + \frac{1}{\sqrt{N}} \left(\Sigma \, \mathrm{d}W\right)_k, \tag{5.15}$$

where W is a D-dimensional vector of independent standard Brownian motions and Σ^2 is given by

$$\Sigma_{lm}^2 = \sum_{j=1}^{M} v_{jl} v_{jm} \int_0^{\tau_0} f_j\left(x\left(t-s\right)\right) \, \mathrm{d}\mu_j\left(s\right).$$

Let ℓ_N denote the stochastic process defined by (5.15).

Theorem 1. For any fixed time $T > 0$, $\Delta\left(N\right) = N^{-1/4}$ is a macroscopic infinitesimal time on $[0, T]$: There exist constants K_1 and K_2, depending only on T and the chemical reaction network, such that

$$P\left(\max_{t\in[0,T]} b_N\left(t\right) - \ell_N\left(t\right) > \frac{K_1}{N^{1/8}}\right) K_2 e^{-\frac{1}{3}N^{1/4}}. \tag{5.16}$$

Theorem 1 has several interesting corollaries. First, for every continuous observable Ψ (real-valued function of trajectories of the processes), the difference between the expected value of Ψ with respect to the dBD and dCLE processes converges to zero as $N \to \infty$. This implies in particular that the distance between every moment of the processes converges to zero as $N \to \infty$. Second, since the right side of (5.16) is summable with respect to N, it follows from the Borel-Cantelli lemma that a trajectory of the dCLE process will, with probability one, lie within the tube of radius $K_1 T\zeta / N^{1/8}$ around the corresponding trajectory of the dBD process for all but finitely many values of N.

One would use Theorem 1 in a concrete situation by first fixing a desirable probability bound, say $1/1000$. Using (5.16), one would then solve for the radius of the tube around the trajectory of the dBD process for which the dCLE trajectory will remain in this tube up to time T with probability at least $999/1000$.

We prove Theorem 1 in three steps (see [37] for a complete proof). First, we prove that the family of measures on trajectories associated with b_N is a tight family, roughly meaning that they are all approximately supported on a common compact set of trajectories.

Second, we prove a version of (5.16) for 'discrete-time tubes' wherein the maximum on the left side of (5.16) is taken not over all $t \in [0, T]$ but rather over only $0, \Delta, 2\Delta, \ldots, (T/\Delta)\Delta$. Third, we study the size of fluctuations of the dBD and dCLE processes on time intervals of length Δ in order to upgrade the discrete-time tube estimate to the continuous-time tube estimate (5.16). We determine the optimal scaling of Δ during this final step. At the beginning of the proof, we assume that Δ scales with N as $\Delta(N) = N^{-\alpha}$ and then look for the optimal value of α. In order to both prove the discrete-time tube estimate and control fluctuations of the dCLE process on intervals of length Δ, we scale the radius $r(N)$ of the continuous-time tube as

$$r(N) = \frac{C_1}{N^{\alpha/2}} + \frac{C_2}{N^{(1-3\alpha)/2}}.$$

The first term in the sum measures local fluctuations of the dCLE while the second comes from the proof of the discrete-time tube estimate. In order to minimize $r(N)$, it is asymptotically optimal to have $\alpha = 1 - 3\alpha$, giving $\alpha = 1/4$.

We conclude this section with a cautionary note about what the main theorem does *not* say. Theorem 1 is formulated for fixed time intervals: one must first fix T and only then allow N to vary. As a consequence, this theorem does not apply when one scales T with N such that $T \to \infty$ as $N \to \infty$, as one does in the large deviations context.

5.3 Discussion

Stochastic differential equations (SDEs) are one of our main tools for modeling noisy processes in nature. Interactions between the components of a system or network are frequently not instantaneous. It is therefore natural to include such delay into corresponding stochastic models. However, the relationship between delay SDEs (dSDEs) and the

processes they model has not been fully established.

Delay stochastic differential equations have previously been formally derived from the delay chemical master equation (dCME). Unlike the chemical master equation, however, the dCME is not closed; this complicates the derivation of dSDE approximations. Closure in this context means the following: Let $P(n,t)$ denote the probability that the stochastic system is in state n at time t. The dCME expresses the time derivative of $P(n,t)$ in terms of joint probabilities of the form $P(j,t;k,t-\tau)$ - the probability that the system is in state j at time t and was in state k at time $t-\tau$, where τ is the delay. The one-point probability distribution $P(\cdot,t)$ is therefore expressed in terms of two-point joint distributions, resulting in a system that is not closed. Timescale separation assumptions have been used to close the dCME. If the delay time is large compared to the other timescales in the system, one may assume that events that occur at time $t-\tau$ are decoupled from those that occur at time t and close the dCME [11, 84] by assuming the joint probabilities may be written as products:

$$P(j,t;k,t-\tau) = P(j,t)\,P(k,t-\tau)\,.$$

Having closed the dCME, one may then derive dSDE approximations [84] as well as useful expressions for autocorrelations and power spectra [11].

Approximations of dSDE type have also been derived using system size expansions such as van Kampen expansions [91, 92] and Kramers-Moyal expansions for both fixed delay [25] and distributed delay [50].

Brett and Galla [12] use the path integral formalism of Martin, Siggia, Rose, Janssen, and de Dominicis to derive the delay chemical Langevin equation (dCLE) without relying on the dCME. Using this formalism, a moment generating functional may be expressed

in terms of the system size parameter N and the sampling rate Δ. In the continuous-time limit, $\Delta \to 0$, the dCLE may be inferred from the moment generating functional. However, the Brett and Galla derivation has some limitations. First, the $\Delta \to 0$ limit cannot be taken without simultaneously letting $N \to \infty$. Intuitively, this is because as $\Delta \to 0$, the Gaussian approximation to the Poisson distribution with mean $N\lambda\Delta$ breaks down unless the parameter $N\lambda$ simultaneously diverges to infinity. Second, the derivation gives no quantitative information about the distance between the dCLE and the original delay birth-death (dBD) process.

This chapter addresses these shortcomings. The dBD process can be approximated by a class of Gaussian processes that includes the dCLE. The proof of this result is addressed in [37]. In particular, for most biophysically relevant propensity functions, the dCLE process will approximate all moments of the dBD process. The proof in [37] includes bounds on the quality of the approximation in terms of the time T for which the approximation is desired to hold and the characteristic system size N (see Theorem 1). The error bounds also indicate that the quality of the dCLE approximation worsens with increasing upper bounds on the reaction propensity functions and state-change vectors. Physically, this means that high reaction rates and reactions that cause large changes in the protein populations are detrimental to the quality of the dCLE approximation.

The dCLE is one of many Gaussian processes that approximate the dBD process. Among all Gaussian approximations with noise components that scale as $1/\sqrt{N}$, the dCLE is optimal because it is the only such approximation that exactly matches the first and second moments of the dBD process. We formally justify this assertion [37] using characteristic functions. As our simulations of the degrade and fire oscillator demonstrate, the dCLE can significantly outperform other Gaussian approximations at moderate

system sizes.

Nevertheless, the quantitative tube estimates in Theorem 1 apply to *any* Gaussian approximation of the dBD process provided the noise scales as $1/\sqrt{N}$. This is significant because it is often advantageous to use linear noise approximations of the dCLE. Delay appears in the drift component of a linear noise approximation but not in the diffusion component. Linear noise approximations are therefore easier to analyze than their dCLE counterparts. In particular, elements of the theory of large deviations for Markovian systems can be extended to SDEs with delay in the drift [75].

For metastable systems, our simulations indicate that the dCLE may capture both temporal information (such as hitting times for the positive feedback model; see Fig. 5.5) and spatial information (such as densities for trajectories corresponding to failed and successful transitions; see Fig. 5.6) for some examples. While SDE approximations (without delay) do not in general capture the dynamics in the presence of metastability even for Markov systems [21, 38, 49], our work suggests that dCLE approximations may be used, on a case by case basis, to study rare events for metastable biochemical systems.

In this work, we have made a number of simplifying assumptions in order to be able to prove the existence of a macroscopic infinitesimal time postulated by Gillespie [29]. In particular we assume the existence of a latent system size parameter N, in terms of which this macroscopic infinitesimal time Δ can be expressed. However, for many biochemical systems, it is often unclear what the system size is, and in this case, a more careful analysis will be required to rigorously justify the use of delay stochastic differential equations to approximate the dynamics of the underlying discrete systems. Further complications can arise when the system size itself changes as a function of time or if there is a scale separation between the fast and slow components of the system. In the non-delayed case, hybrid

systems [39,70,71] are often used to address these challenges. Developing a mathematical theory for hybrid systems in the presence of delays is an important future direction.

We have shown that the dCLE provides an accurate approximation of a number of stochastic processes. Although we chose gene regulatory networks in our examples, the theory is applicable to general birth-death processes with delayed events. SDEs, and the chemical Langevin equation in particular, are fundamental in modeling and understanding the behavior of natural and engineered systems. We therefore expect that the dCLE will be widely applicable when delays impact system dynamics.

Chapter 6

Conclusion and future work

Protein production is a complex processes that requires hundreds or thousands of reactions before a mature protein is produced. Here we quantified delay by assuming that a single mRNA gives rise to a protein in a random amount of time. This allowed us to simplify the protein production process and examine gene regulatory networks. Our main focus was to explore the effects of distributed delay on simple gene networks and bistable networks. To do so we used an extension of the stochastic simulation algorithm which accounts for delay. Mathematical analysis was performed with tractable examples and verified via simulations.

We found that in a transcriptionally regulated signaling process, the mean signaling time decreases with increasing variability in the delay time. Similarly, an increase in delay variability results in a decrease in the mean peak height (maximum number of proteins in a cycle) in the negative feedback oscillator. In the case of bistable switches, a small increase in fixed delay resulted in an increase in stability. Finally, we showed that that using the correct dCLE version which includes delay in its drift and diffusion term we

can approximate the dSSA even for relatively small system sizes.

These results can give us valuable insights on the gene regulatory networks dynamics. It is not yet clear however, if this type of modeling captures the main mechanisms that affect the stability and dynamics of gene regulatory networks. One potential problem is that mRNA and hence protein production is bursty. Moreover, each mRNA is translated into a number of proteins, this number can range from 1 to dozens. In most of our simulations we assumed that there is an infinite number of ribosomes ready to begin translation of mRNA as soon as this becomes available. All of these could affect the properties of the stochastic properties of signaling in gene networks, and hence their dynamics.

Our simulations account for cells that have a constant volume. In most of the cases however, cells grow and undergo divisions. It is known that cell division occurs at a shorter time scale that protein degradation. A new class of simulation algorithms that account for cell division and delay in protein production could be used to model these effects more accurately.

We must keep in open mind however when trying to understand the results obtained from their simulations and not reach conclusions hastily. Preliminary results seem to point out that in the case of bistable switches, a delayed stochastic simulation algorithm that accounts for cell division events will result in a decrease in bistability. Further analysis needs to be performed in bistable switches under a cell division simulation algorithm with perhaps even distributed delay as opposed to fixed delayed as we have done previously.

Appendix

The content that follows is a collection of problems that appeared during the completion of the several projects discussed. Here we assume that the reader is familiar with the Linux/Unix operating system and basic use of the Terminal.

A File Transfer

We begin by describing a situation which many of us encounter daily: data transfer. During the creation of the simulations done for this work we worked mainly on our personal computers which have less computing power when compared to the ones accessible in other networks or clusters.

A typical problem occurs when we have, for instance, a MATLAB script that needs to be run in a remote machine. A naive solution is to use the `scp` command. This will allow us to copy the file to a remote location with the help of `ssh`.

```
scp myscript.m username@servername:~/path/in/to/directory/myscript.m
```

This method may work initially even with a large file collection. Eventually we will realize that not all the files need to be copied, only a few.

A.1 One-way synchronization

The `rsync` [88] tool allows us to copy large numbers of files in the same fashion as `scp` with the added benefit that if a copy already exists in the remote directory then it will only update the files that are different.

```
rsync -razv local_directory/ username@servername:~/remote_directory/
```

The above command copies the contents of `local_directory` and syncs them with the contents of `remote_directory` in the remote machine. We must warn though that `rsync` is a one-way synchronization tool. That is, if we happen to work in a remote machine and modify one of the files that we sync regularly, we must first reverse the synchronization. We should execute

```
rsync -razv username@servername:~/remote_directory/ local_directory/
```

from our local machine, otherwise we will lose all the changes made to the file in the remote machine. Beginners will quickly find out that with `rsync` it is difficult track which files were modified and may overwrite their work.

A common working routine goes as follows: We work from our home computer, and at the end of our shift we synchronize the files to the remote server. Once we reach the office, we use their machines and continue working with the code and data which was sent from our home computer. When we return to our home computer, we must to retrieve our

work from the remote machine at our office before resuming work at home. If we forget to retrieve our work from the office then we will end up with conflicting copies of the files.

A.2 Two-way synchronization

To avoid the need to track synchronization direction we can use `unison` [68]. This software works similarly to `rsync` but it provides a two-way synchronization. However, it still cannot solve all the problems that `rsync` fails to solve if we do perform regular synchronizations. There may be times when file conflicts are encountered. For instance, a file was modified in machine *A* and no synchronization was performed. Later, in machine *B*, a modification of the file occurs. Now there are two copies of a file and `unison` will have trouble deciding which copy to keep. In this case `unison` makes a back up copy if you had previously specified in its configuration.

Another possible solution is to use the Dropbox service. When installed in many machines we do not have to worry about having conflicting files as long as we are the only user with access to them. Eventually, we may need to collaborate with others. The pros of Dropbox are that it allows us to share files and directories with our collaborators and it creates regular back-ups of our files. Possible cons are that these files are now the property of a company (not on your server), and that conflicting copies of files can arise when you allow collaborators to view and edit your files.

A.3 Repositories

There exist several free online services that provide access to repositories. They allow us to store any type of digital data and keep several versions of the project. One service in particular is Github [69]. Github uses the `git` software [87] which was developed by Linus Torvalds. The software itself has a steep learning curve, but to know that our projects are securely stored makes the trouble of learning its use worth it.

A typical workflow with repositories goes as follows: First we assume that a `git` repository exists in a remote server and that we have a clone of this repository in computer A and computer B. Suppose we work on computer A. At the end of the session we must commit our work and push the changes made to the repository in the remote server. Then if we wish to work in computer B, we must pull back those changes. In case we forget to pull the changes and we accidentally work on the same files that were modified while in computer A, then `git` will warn us of the possible conflicts the next time we try to pull the changes on computer B. This method surpasses Dropbox since we are also able to see the changes to the documents we are working on. This is useful when working with several collaborators.

Github is a very robust service which connects many developers to create projects. Unfortunately all of the projects are made public by default unless private repositories are purchased for a fee. There are projects which behave like Github. Two examples are Gitolite and Gitosis. The difference is that these projects allow us to create our own repositories in our own private servers.

Unfortunately both Gitolite and Gitosis assume that the server can be accessed by a common user which is called 'git' and requires us to set up git as a server. To do so we

must be an administrator of the server. This would work for private companies who desire to keep their projects protected from the eyes of the public. We have developed a Python package called Promus to address this issue.

A.4 Promus

Promus is a solution that came to be after exploring the possibilities of `ssh` and `git`. As with Gitolite and Gitosis, promus allows users to access a remote repository. The main difference is that promus can work for any user without the need of the server administrator.

Promus is a Python script written to facilite the interations between `ssh` and `git`. The first version of promus allowed us to develop code and document drafts in our own private repositories. It was developed as a tool which members of academia can use to work on their projects or to store several versions of their grade sheets as the semester progresses.

This project is ongoing and is maintained at `https://github.com/jmlopez-rod/promus`. To find out more about Promus you may visit the Github page or visit its documentation site `http://promus.readthedocs.org`.

B Calling C++ from scripting languages

Scripting languages such as Python and MATLAB allow users to develop ideas in a relatively short amount of time when compared to languages such as C++. The tradeoff is that since Python and MATLAB scripts are interpreted they run much slower than a compiled C++ file.

The first versions of the SSA without delay used in our studies were written in MATLAB. Eventually the algorithms were written in C++ due to the speed increase obtained. Most of the data obtained from C++ had to be organized to fulfill specific functions. For instance, when obtaining realizations for a model, we may wish to obtain ordered pairs of the time and the state of the system. A simple file with rows containing this information suffices. These files can then be read from MATLAB or Python and analysis can then begin.

B.1 Data Serialization

MATLAB and Python both have a way of serializing their internal structures. In Python you can "pickle" objects into a file while in MATLAB you simply "save" objects to a file. The storage of data structures into a sequence of bytes is called data serialization. Many data files of different structures arose from the C++ executables along with MATLAB scripts to read their contents. After exploring the C++ language and realizing that all data structures are made up of three basic objects it was not hard to realize how to create a simple serializer in C++.

The idea behind this serializer is that the data stored in a file should describe itself. For instance, if we wanted to store an integer and a real number in a file we could possibly have the following:

```
−1 3.14
```

Note, however, that this is not enough information to completely describe the original C++ object. We know that since -1 is an integer it must have been stored in an `int` type

object. There are however other types: `short int`, `long int`. The difference between these types is the number of bytes each one requires. The same goes for the number 3.14: there is no way of knowing if this number was stored in a `float` or `double`. A solution to store all the information would be to notice that a machine can represent integers (I) and real numbers (R) of different sizes. The following file contains all the information required to load some variables in a scripting language.

```
2
a I 4 -1
b R 8 3.14
```

This file states that there are two variables. The first one is an integer of four bytes holding the value of -1 and should be assigned the name *a* when loaded. The second variable is a real number of eight bytes holding the value of 3.14 and should be named *b*. This is a very simple example but it contains the basic idea that allowed us to pipe the C++ output directly into the interpreters instead of writing data files.

B.2 Excentury

Given that there is a way of reading and writing any type of data to a file in the same format across C++ and scripting languages, now we develop a file format that will allow us to call C++ code from the interpreters. The key realization is that we are interested in calling C++ functions. Each scripting language usually has a way of calling compiled C++ code. In the case of MATLAB, we may create mex files. For python we can create a dynamic library which can be called from the Python interpreter.

Each language has its specifics on how to call C++ code. Excentury removes these

specifications by processing a simple formatted file which it then transforms into a mex file so that the code may be called from MATLAB or to another C++ file format so that it may be called from Python. Consider the following file named `tools.xcpp`:

```
"""Tools

This file contains an implementation of Newton's method.

"""

-----------------------------------------------------------------------

@def{square_root}
    """Compute the square root of a number using Newton's method."""
    @param{double, a(2), "the input to the square root function"}
    @param{double, x0(1), "initial guess"}
    @param{int, iter(10), "number of iterations"}
@body[[
    if (a < 0) {
        excentury::error("input 'a' must be non-negative");
    }
    double x = x0;
    for (int i=0; i < iter; ++i) {
        x = x - (x*x - a)/(2.0*x);
    }
]]
@ret[[
    @ret{x, "x"}
]]
-----------------------------------------------------------------------
```

Its format is simple. We start by defining a function by using the keyword "@def", then we provide a documentation string followed by the parameters of the function. What follows is the body of the function; this code is pure C++ but notice how we can use the parameters we had previously specified. Finally we have a return block which specifies what values should be returned. This file contains all the essential information we need to create a mex file or some other C++ file for Python. By building Excentury we have taken the interoperability knowledge into one package. Excentury allows us to quickly and effortlessly call the C++ function declared in `tools.xcpp` from Python by using

```
xcpp tools.xcpp to python
```

Similarly, to call from MATLAB we execute

```
xcpp tools.xcpp to matlab
```

Excentury has become the core and format for all of our latest simulations in our projects. You may obtain more information on this ongoing project by visiting its Github page `https://github.com/jmlopez-rod/excentury`.

C Assist

ASSIsT is an acronym for "a stochastic simulation toolkit". This project is a collection of all the methods along with models and data collectors used for the simulations in this work. The toolkit is modular; you may create new models and use existing methods and data collectors provided by the library to quickly create a simulation.

This project is in its early stages and will be updated on its Github page. You may visit `https://github.com/jmlopez-rod/assist` to see updates and other examples that explain how the simulation results from this work were obtained.

Bibliography

[1] U. Alon. Network motifs: theory and experimental approaches. *Nat Rev Genet*, 8:450–461, 2007.

[2] A. Amir, O. Kobiler, A. Rokney, A. B. Oppenheim, and J. Stavans. Noise in timing and precision of gene activities in a genetic cascade. *Molec Syst Biol*, 3:71, 2007.

[3] A. Amir, S. Meshner, T. Beatus, and J. Stavans. Damped oscillations in the adaptive response of the iron homeostasis network in *E. coli*. *Molec Microbiol*, 76:428–436, 2010.

[4] A. Arazi, E. Ben-Jacob, and U. Yechiali. Bridging genetic networks and queueing theory. *Physica A*, 332:585–616, 2004.

[5] E. Aurell and K. Sneppen. Epigenetics as a first exit problem. *Phys Rev Lett*, 88:048101, 2002.

[6] N. Q. Balaban, J. Merrin, R. Chait, L. Kowalik, and S. Leibler. Bacterial persistence as a phenotypic switch. *Science*, 305:1622–1625, 2004.

[7] Basil Bayati, Philippe Chatelain, and Petros Koumoutsakos. D-leaping: Accelerating stochastic simulation algorithms for reactions with delays. *Journal of Computational Physics*, 228(16):5908–5916, 2009.

[8] G. Bel, M. Munsky, and I. Nemenman. The simplicity of completion time distributions for common complex biochemical processes. *Phys Biol*, 7:016003, 2010.

[9] M. R. Bennett, D. Volfson, L. Tsimring, and J. Hasty. Transient dynamics of genetic regulatory networks. *Biophys J*, 92(10):3501–3512, 2007.

[10] D. Bratsun, D. Volfson, L. S. Tsimring, and J. Hasty. Delay-induced stochastic oscillations in gene regulation. *Proc Natl Acad Sci USA*, 102(41):14593–14598, 2005.

[11] Dmitri Bratsun, Dmitri Volfson, Lev S. Tsimring, and Jeff Hasty. Delay-induced stochastic oscillations in gene regulation. *Proceedings of the National Academy of Sciences of the United States of America*, 102(41):14593–14598, 2005.

[12] Tobias Brett and Tobias Galla. Stochastic processes with distributed delays: chemical langevin equation and linear-noise approximation. `http://arxiv.org/abs/1302.7166`, 2013.

[13] R. Bundschuh, F. Hayot, and C. Jayaprakash. Fluctuations and slow variables in genetic networks. *Biophys J*, 84:1606–1615, 2003.

[14] X. Cai and Z. Xu. K-leap method for accelerating stochastic simulation of coupled chemical reactions. *Journal of Chemical Physics*, 126(7), 2007.

[15] Xiaodong Cai. Exact stochastic simulation of coupled chemical reactions with delays. *The Journal of Chemical Physics*, 126(124108), 2007.

[16] Y. Cao, D.T. Gillespie, and L.R. Petzold. Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics*, 124(4), 2006.

[17] Y. Cao, D.T. Gillespie, and L.R. Petzold. Adaptive explicit-implicit tau-leaping method with automatic tau selection. *Journal of Chemical Physics*, 126(22), 2007.

[18] K. C. Chen, A. Csikasz-Nagy, B. Gyorffy, J. Val, B. Novak, and J. J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molec Biol Cell*, 11:369–391, 2000.

[19] L. Chen and K. Aihara. Stability of genetic regulatory networks with time delay. *IEEE Trans Circ Syst*, 49:602–608, 2002.

[20] H. De Jong. Modeling and simulation of genetic regulatory systems: A literature review. *J Comp Biol*, 9:67–103, 2002.

[21] Charles R. Doering, Khachik V. Sargsyan, and Leonard M. Sander. Extinction times for birth-death processes: exact results, continuum asymptotics, and the failure of the Fokker-Planck approximation. *Multiscale Model. Simul.*, 3(2):283–299 (electronic), 2005.

[22] 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.

[23] Stewart N. Ethier and Thomas G. Kurtz. *Markov processes*. Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics. John Wiley & Sons Inc., New York, 1986. Characterization and convergence.

[24] Markus Fischer and Peter Imkeller. A two-state model for noise-induced resonance in bistable systems with delay. *Stoch. Dyn.*, 5(2):247–270, 2005.

[25] Tobias Galla. Intrinsic fluctuations in stochastic delay systems: Theoretical description and application to a simple model of gene regulation. *Phys. Rev. E*, 80:021909, Aug 2009.

[26] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, 2000.

[27] Michael A. Gibson and Jehoshua Bruck. Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, March 2000.

[28] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J Phys Chem*, 81:2340–2361, 1977.

[29] Daniel T Gillespie. The chemical langevin equation. *The Journal of Chemical Physics*, 113(1):297–306, 2000.

[30] D.T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[31] D.T. Gillespie, A. Hellander, and L.R. Petzold. Perspective: Stochastic algorithms for chemical kinetics. *Journal of Chemical Physics*, 138(17), 2013.

[32] L. Goentoro, O. Shoval, M. W. Kirschner, and U. Alon. The incoherent feedforward loop can provide fold-change detection in gene regulation. *Mol Cell*, 36:894–899, 2009.

[33] I. Golding, J. Paulsson, S. M. Zawilski, and E. C. Cox. Real-time kinetics of gene activity in individual bacteria. *Cell*, 123:1025–1036, 2005.

[34] B. C. Goodwin. Oscillatory behavior in enzymatic control processes. *Adv Enzyme Regul*, 3:425–438, 1965.

[35] A. Grönlund, P. Lötstedt, and J. Elf. Costs and constraints from time-delayed feedback in small regulatory motifs. *Proc Natl Acad Sci USA*, 107:8171–8176, 2010.

[36] D. Gross, J. Shortle, J. Thompson, and C. Harris. *Fundamentals of queueing theory*. John Wiley & Sons Inc., Hoboken, New Jersey, fourth edition, 2008.

[37] Chinmaya Gupta, José Manuel López, Robert Azencott, Matthew R. Bennett, Krešimir Josić, and William Ott. Supplementary material. `ftp://ftp.aip.org/epaps/journ_chem_phys/E-JCPSA6-140-038420/`, 2013.

[38] Peter Hanggi, Hermann Grabert, Peter Talkner, and Harry Thomas. Bistable systems: master equation versus Fokker-Planck modeling. *Phys. Rev. A (3)*, 29(1):371–378, 1984.

[39] Eric L Haseltine and James B Rawlings. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *The Journal of chemical physics*, 117(15):6959–6969, 2002.

[40] E. He, O. Kapuy, R. A. Oliveira, F. Uhlmann, J. J. Tyson, and B. Novák. Systems-level feedbacks make the anaphase switch irreversible. *Proc Natl Acad Sci USA*, 108:10016–10021, 2011.

[41] Desmond J. Higham. Modeling and simulating chemical reactions. *SIAM Rev.*, 50(2):347–368, 2008.

[42] T. Hong, J. Xing, L. Li, and J. J. Tyson. A simple theoretical framework for understanding heterogeneous differentiation of cd4$^+$ t cells. *BMC Syst Biol*, 6:66, 2012.

[43] S. Hooshangi, S. Thiberge, and R. Weiss. Ultrasensitivity and noise propogation in a synthetic transcriptional cascade. *Proc Natl Acad Sci USA*, 102:3581–3586, 2005.

[44] F. Jacob and J. Monod. Genetic regulatory mechanisms in synthesis of proteins. *J Mol Biol*, 3:318–356, 1961.

[45] Zheng-Lin Jia. Numerical investigation of noise enhanced stability phenomenon in a time-delayed metastable system. *Chin. Phys. Lett.*, 25(4):1209, 2008.

[46] M. Kaern, W. J. Blake, and J. J. Collins. The engineering of gene regulatory networks. *Annual Review of Biomedical Engineering*, 5:179–206, 2003.

[47] T. B. Kepler and T. C. Elston. Stochasticity in transcriptional regulation: origins consequences, and mathematical representations. *Biophys J*, 81:3116–3136, 2001.

[48] Donald E Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley Professional, 2014.

[49] Harold J. Kushner. Robustness and approximation of escape times and large deviations estimates for systems with small noise effects. *SIAM J. Appl. Math.*, 44(1):160–182, 1984.

[50] L. F. Lafuerza and R. Toral. Role of delay in the stochastic creation process. *Phys. Rev. E*, 84:021128, Aug 2011.

[51] Y. H. Lan, T. C. Elston, and G. A. Papoian. Elimination of fast variables in chemical Langevin equations. *J Chem Phys*, 129:214115, 2008.

[52] D. R. Larson, D. Zenklusen, B. Wu, J. A. Chao, and R. H. Singer. Real-time observation of transcription initiation and elongation on an endogenous yeast gene. *Science*, 332:475–478, 2011.

[53] Jinzhi Lei, Guowei He, Haoping Liu, and Qing Nie. A delay model for noise-induced bi-directional switching. *Nonlinearity*, 22(12):2845–2859, 2009.

[54] E. Levine and T. Hwa. Stochastic fluctuations in metabolic pathways. *Proc Natl Acad Sci USA*, 104:9224–9229, 2007.

[55] J. Lewis. Autoinhibition with transcriptional delay: A simple mechanism for the zebrafish somitogeneis oscillator. *Curr Biol*, 13:1398–1408, 2003.

[56] J. M. Mahaffy and C. V. Pao. Models of genetic control by repression with time delays and spatial effects. *J Math Biol*, 20:39–57, 1984.

[57] R. Maithreye, R. R. Sarkar, V. Parnaik, and S. Sinha. Delay-induced transient increase and heterogeneity in gene expression in negatively auto-regulated gene circuits. *PLoS One*, 3:e2972, 2008.

[58] S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *Proc Natl Acad Sci USA*, 100:11980–11985, 2003.

[59] S. Mangan, S. Itzkovitz, A. Zaslaver, and U. Alon. The incoherent feed-forward loop accelerates the response-time of the *gal* system in *escherichia coli*. *J Mol Biol*, 356:1073–1081, 2006.

[60] C. Masoller. Distribution of Residence Times of Time-Delayed Bistable Systems Driven by Noise. *Phys. Rev. Lett.*, 90(2):020601, 2003.

[61] W. Mather, M. R. Bennett, J. Hasty, and L. S. Tsimring. Delay-induced degrade-and-fire oscillations in small genetic circuits. *Phys Rev Lett*, 102(6):068105, 2009.

[62] W. Mather, N. A. Cookson, J. Hasty, L. S. Tsimring, and R. J. Williams. Correlation resonance generated by coupled enzymatic processing. *Biophys J*, 99:3172–3181, 2010.

[63] William Mather, Matthew R. Bennett, Jeff Hasty, and Lev S. Tsimring. Delay-induced degrade-and-fire oscillations in small genetic circuits. *Phys. Rev. Lett.*, 102:068105, Feb 2009.

[64] H. H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269:650–656, 1995.

[65] N. A. Monk. Oscillatory expression of Hes1, p53, and NF-κB driven by transcriptional time delays. *Curr Biol*, 13(16):1409–1413, 2003.

[66] D. Nevozhay, R. M. Adams, E. Van Itallie, M. R. Bennett, and G. Balázsi. Mapping the environmental fitness landscape of a synthetic gene circuit. *PLoS Comp Biol*, 8:e1002480, 2012.

[67] E. M. Ozbudak, M. Thattai, H. N. Lim, B. I.. Shraiman, and A. van Oudenaarden. Multistability in the lactose utilization network of *Escherichia coli*. *Nature*, 427:737–740, 2004.

[68] Benjamin C. Pierce. Unison file synchronizer. `http://www.cis.upenn.edu/~bcpierce/unison/`, 2009.

[69] Tom Preston-Werner, Chris Wanstrath, and PJ Hyett. Github. `https://github.com/about`, 2014.

[70] Howard Salis and Yiannis Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *The Journal of chemical physics*, 122(5):054103, 2005.

[71] Howard Salis, Vassilios Sotiropoulos, and Yiannis N Kaznessis. Multiscale hy3s: Hybrid stochastic simulation for supercomputers. *BMC bioinformatics*, 7(1):93, 2006.

[72] R. Schlicht and G. Winkler. A delay stochastic process with applications in molecular biology. *J Math Biol*, 57:613–648, 2008.

[73] Robert Schlicht and Gerhard Winkler. A delay stochastic process with applications in molecular biology. *J. Math. Biol.*, 57(5):613–648, 2008.

[74] Ira B Schwartz, Thomas W Carr, Lora Billings, and Mark Dykman. Noise induced switching in delayed systems. *arXiv preprint arXiv:1207.7278*, 2012.

[75] Ira B Schwartz, Thomas W Carr, Lora Billings, and Mark Dykman. Noise induced switching in delayed systems. *arXiv preprint arXiv:1207.7278*, 2012.

[76] M. Scott. Long delay times in reaction rates increase intrinsic fluctuations. *Phys Rev E*, 80:031129, 2009.

[77] V. Sevim, X. W. Gong, and J. E. S. Socolar. Reliability of transcriptional cycles and the yeast cell-cycle oscillator. *PLoS Comp Biol*, 6(7):e1000842, 2010.

[78] P. Smolen, D. A. Baxter, and J. H. Byrne. Effects of macromolecular transport and stochastic fluctuations on dynamics of genetic regulatory systems. *Am J Physiol Cell Physiol*, 277(4 Pt 1):C777–C790, 1999.

[79] P. Smolen, D. A. Baxter, and J. H. Byrne. A reduced model clarifies the role of feedback loops and time delays in the drosophila circadian oscillator. *Biophys J*, 83(5):2349–2359, 2002.

[80] K. Sriram and M. S. Gopinathan. A two variable delay model for the circadian rhythm of *Neurospora crassa*. *J Theor Biol*, 231:23–38, 2004.

[81] J. Stricker, S. Cookson, M. R. Bennett, W. H. Mather, L. S. Tsimring, and J. Hasty. A fast, robust and tunable synthetic gene oscillator. *Nature*, 456(7221):516–519, 2008.

[82] M. Thattai and A. van Oudenaarden. Attenuation of noise in ultrasensitive signaling cascades. *Biophys J*, 82:2943–2950, 2002.

[83] Tianhai Tian, Kevin Burrage, Pamela M. Burrage, and Margherita Carletti. Stochastic delay differential equations for genetic regulatory networks. *J. Comput. Appl. Math.*, 205(2):696–707, 2007.

[84] Tianhai Tian, Kevin Burrage, Pamela M. Burrage, and Margherita Carletti. Stochastic delay differential equations for genetic regulatory networks. *Journal of Computational and Applied Mathematics*, 205(2):696–707, 2007. Special issue on evolutionary problems.

[85] G. Tiana, M. H. Jensen, and K. Sneppen. Time delay as a key to apoptosis induction in the p53 network. *Eur Phys J*, 29:135–140, 2002.

[86] M. Tigges, T. T. Marquez-Lago, J. Stelling, and M. Fussenegger. A tunable synthetic mammalian oscillator. *Nature*, 457(7227):309–312, 2009.

[87] Linus Torvalds. git. `http://git-scm.com/`, 2014.

[88] Andrew Tridgell and Wayne Davison. rsync. `http://rsync.samba.org/`, 2014.

[89] L. Tsimring and a. Pikovsky. Noise-Induced Dynamics in Bistable Systems with Delay. *Phys. Rev. Lett.*, 87(25):1–4, 2001.

[90] M. Ukai-Tadenuma, R. G. Yamada, H. Xu, J. A. Ripperger, A. C. Liu, and H. R. Ueda. Delay in feedback repression by *Cryptochrome 1* is required for circadian clock function. *Cell*, 144:268–281, 2011.

[91] N. G. van Kampen. A power series expansion of the master equation. *Canadian Journal of Physics*, 39(4):551–567, 1961.

[92] N. G. van Kampen. *Stochastic processes in physics and chemistry*, volume 1. Elsevier, 1992.

[93] Can-Jun Wang. Delays Induce Different Switch in a Stochastic Single Genetic Regulation System With a Positive Autoregulatory Feedback Loop. *Int. J. Mod Phys B*, 27(11):1350085, 2013.

[94] Canjun Wang, Ming Yi, and Keli Yang. Time delay-accelerated transition of gene switch and -enhanced stochastic resonance in a bistable gene regulatory model. In *Systems Biology (ISB), 2011 IEEE International Conference on*, pages 101–110, 2011.

[95] P. B. Warren and P. R. ten Wolde. Chemical models of genetic toggle switches. *J Phys Chem B*, 109:6812–6823, 2005.

[96] Z. Xu and X. Cai. Unbiased -leap methods for stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 128(15), 2008.

[97] R. Zhu, A. S. Ribeiro, D. Salahub, and S. A. Kauffman. Studying genetic regulatory networks at the molecular level: Delayed reaction stochastic models. *J Theor Biol*, 246:725–745, 2007.

[98] R. Zhu and D. Salahub. Delay stochastic simulation of single-gene expression reveals detailed relationship between protein noise and mean abundance. *FEBS Lett*, 582:2905–2910, 2008.