Application, Computation, and Theory for Synthetic Gene Circuits

Thesis by

Anandh Swaminathan

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Caltech

California Institute of Technology

Pasadena, California

2018

(Defended August 31, 2017)

Acknowledgments

I would like to start off by acknowledging my committee members for their helpful feedback and positive support. I would like to thank Lea Goentoro for her positive attitude and boundless curiosity, both of which have rubbed off on me to some degree. I'd like to thank John Doyle for his terse but profound comments, many of which I did not understand until years after the fact. I'd like thank Jim Beck for his enthusiasm in discussing and teaching parameter estimation and Markov chain Monte Carlo. I'd like to thank Michael for being the authoritative source on how to give presentations, and for showing me that science is as much about art as it is about math.

In addition to my committee members, I am hugely grateful to Justin Bois and Yihan Lin for their impact on my graduate career. I learned a lot from TA'ing for Justin's systems biology course. From Justin, I learned how to approach research problems and how to structure my research workflow, through essentially complete imitation. Yihan taught me everything I know about doing biology in the lab, including the most important skill, which is to believe that the next thing you are going to do is always going to work.

I am also thankful to have been around a great group of friends and colleagues throughout my time here. I'd like to thank the past and present members of the Murray Lab for making graduate school a more fun experience. I would like to especially thank Victoria Hsiao and Marcella Gomez for not only their friendship, but also their role as my informal advisors and collaborators over the last two years. I want to thank Anu Thubagere and Vanessa Jonsson for their support as more senior members in my lab. I would also like to thank Enoch Yeung and Yutaka Hori for their guidance and advice. A special thanks goes to Vipul Singhal, with whom I have shared a lot of my personal and professional experiences over the last six years.

I thank my friend Gautham Sridharan, who reminded me that the best way to accomplish any large task is to do what Tom Brady and Bill Belichick would do: take it one day at a time.

I have to also thank my class in the CDS program, Seungil You, Kevin Shan, and Ivan Papusha, without whom I would never have passed the qualifying exam or understood anything about control theory. All three have been wonderful friends as well.

I also am indebted to my friends who are not colleagues and with whom I almost never discuss research. I'd like to thank Karthik Seetharam for being a great roommate despite being absurdly messy. I'd like to thank Dennis Kim, with whom I became friends instantly during orientation upon finding we were both from Massachusetts. I am tremendously grateful to Fadl Saadi for finding and helping me procure an Apple Pencil. I'd like to also thank Linhan Shen for a wonderful collection of Patriots memorabilia. A big thanks to David Chen and the rest of my intramural basketball team for a dominant run over the last four years. Graduate school would have also been much more dull without the time spent with Bassam Helou, Vipul Singhal, Eric Wolff, and many others.

Penultimately, I have to acknowledge my advisor Richard Murray. Richard is one of those rare people who despite being incredibly successful professionally is an even better person. He was patient with me throughout the early part of my PhD while I was struggling with research and offered nothing but support. While many professors prioritize accomplishing research goals above anything else, I have always believed Richard's top priority is the personal development of his students.

Finally, and most importantly, I have to thank my family, Krupa, Vidhya, Raja, and Juno, without whom none of my past, present, or future accomplishments in life would be possible.

Abstract

The field of synthetic gene circuits is concerned with engineering novel gene expression dynamics into organisms. This field, a subset of synthetic biology, was started almost two decades ago with the creation of two synthetic circuits: a bistable toggle switch and an oscillator. From the very outset, modeling has played a role in the development of synthetic circuits. However, modeling has been used to gain qualitative insight into dynamics, and actual quantitative modeling has been lagging behind.

Parameters for quantitative models for biological systems often cannot be adequately estimated from measured data, because far too many sets of parameters can produce the same set of limited measured outputs. Additionally, models for synthetic gene circuits are often not correct the first time, and iterating on cycles of modeling and parameter estimation is difficult. Finally, there is a gap between development of modeling and system identification tools and their application to experiments on actual synthetic gene circuits.

This thesis attempts to work towards addressing these issues with quantitative modeling for synthetic gene circuits. First, we derive theoretical conditions for identifiability of stochastic linear systems from heterogenous types of measurement data. These identifiability conditions can provide insight into what type of model to use and what measurements to collect in order to ensure that the resulting model can be identified.

Second, we develop a software package for fast and flexible modeling and parameter estimation for synthetic gene circuits. The user can input models into our software using a simple text format and perform simulations of all types at optimized speeds. By inputting measured experimental data along with the model, the software can be used to perform Bayesian parameter estimation in an automated manner. To bridge the gap between computation and application, we apply this software to parameter estimation of DNA recombinase dynamics using real experimental data collected in an *in vitro* cell extract.

Finally, we use modeling to guide the design of an improved single gene synthetic oscillator. While the original synthetic genetic oscillator contained three genes, we show that a simple circuit with a single gene can produce robust and synchronized oscillations across a population.

Our results constitute an additional step towards the incorporation of quantitative modeling and parameter inference as part of the design-build-test cycle for synthetic gene circuits.

Published Content and Contributions

V. Hsiao, A. Swaminathan, and R. M. Murray, ”Control theory for synthetic biology,” To appear, *IEEE Control Systems Magazine*.

A. Swaminathan wrote approximately half of the paper, developed numerical examples, and participated in proofreading and revising the paper.

A. Swaminathan, V. Hsiao, and R. M. Murray, ”Quantitative modeling of integrase dynamics using a novel Python toolbox for parameter inference in synthetic biology,” *bioRxiv*, 2017, doi: *10.1101/121152*.

A. Swaminathan conceived the project, wrote the software, performed parameter inference, and wrote the manuscript.

A. Swaminathan and R. M. Murray, ”Linear system identifiability from distributional and time series data,” in *American Control Conference (ACC)*, 2016, pp. 392 – 399, doi: *10.1109/ACC.2016.7524946*.

A. Swaminathan conceived the project, performed the research, and wrote the manuscript.

Contents

Chapter 1

Introduction and Background

The field of synthetic biology consists of engineering novel functionality into organisms such as bacteria, yeast, plants, or mammalian cells. This novel functionality can have many possible applications. Engineered organisms have potential applications as diagnostic tools in medicine [52] and as micro-scale factories for chemical production [45], among many others.

Specifically within synthetic biology, which can intersect with many other fields such as protein design, metabolic engineering, and medicine, the field of synthetic gene circuits is concerned with engineering novel dynamics and computation into organisms. The idea is that these dynamics and small scale computations can be combined as a part of a larger synthetic program that uses organisms to solve a specific problem. For example, an engineered gut bacteria that treats diabetes may be able to produce insulin, but it should only do so when blood sugar increases.

The two original synthetic gene circuits were an oscillator [19] and a bistable toggle switch [23]. In the oscillator, three genes each repress the production of the next gene in a cyclic fashion, which generates oscillations in the expression of each gene and also a fluorescent reporter protein. In the toggle switch, two genes that each repress each other's production are used to create two stable steady states. When the first gene is high, it will repress the second gene and keep the second gene low, and vice versa. Other synthetic gene circuits include logic gates [62], state machines [36, 77], and feedback control circuits [35].

Modeling can guide the design-build-test cycle of synthetic gene circuits. Modeling was used to aid in designing the original oscillator and toggle switch in order to select genetic parts that were more likely to display the observed behaviors. Additionally, modeling can sometimes predict how to improve a poorly functioning gene circuit. However, in general, the application of models in synthetic biology has been qualitative. That is, models are a useful tool that can be used to gain insight into the system at hand, but the actual quantitative numbers produced by models are often not as important.

Chapter 2 of this thesis contains a review of the different types of measurements, models, and system identification techniques used in synthetic biology [37].

This thesis mainly attempts to add to the field of synthetic biology in three areas: application, computation, and theory, specifically by considering modeling in each area.

On the theory side, parameters for biological models are often not identifiable and measurements on biological systems can provide heterogenous types of data. In Chapter 3, we formulate and solve an identifiability problem for stochastic linear systems given heterogenous types of data [88]. Such a result provides insight into the types of measurements required to perform parameter estimation on a real gene circuit.

In Chapter 4, we cover computation for synthetic gene circuits. We develop a software package that can read in user specified or industry standard models and simulate the models in a wide variety of ways. The software can generate deterministic or stochastic data and has been optimized to run fast. The speed is necessary for the software's main purpose, which is to perform Bayesian parameter estimation for user defined models of gene circuits given actual measured data. In order to highlight the use of this software, we use the software to identify dynamic parameters for DNA recombinases in an in vitro extract [87].

Finally, in Chapter 5, we use modeling to guide the improvement of a single gene oscillator. We show that by simply changing one part and tuning DNA copy numbers, we can vastly improve the oscillatory performance of this simple gene circuit in both a deterministic and stochastic sense. The results suggest that even the simplest gene circuits can produce robust dynamics if the parts are tuned appropriately.

Chapter 2

A Review of Modeling and System Identification for Synthetic Gene Circuits

System identification is particularly challenging for biological systems because a majority of the processes that occur within the organism are part of the underlying host biology, which may not be well understood and also difficult to measure. This can result in severe limitations on creating accurate models of synthetic gene circuits that can be used for control design or other computational explorations of circuit behavior. Considerations for model choice and methodology for system identification are largely dependent on which types of data can be collected.

Here, we describe the types of measurements and classes of models that are commonly used to quantitatively model synthetic gene circuits. We then summarize the field of system identification for synthetic gene circuits, first by focusing on deterministic identification, and second by focusing on stochastic identification from population snapshots and time series data respectively. Figure 2.1 serves as a pictorial comparison of deterministic and stochastic identification techniques.

2.1 Measurement Types for Synthetic Gene Circuits

There are three main types of data that can be collected on synthetic gene circuits operating *in vivo*: bulk data, single cell population snapshot data, and time series single cell data.

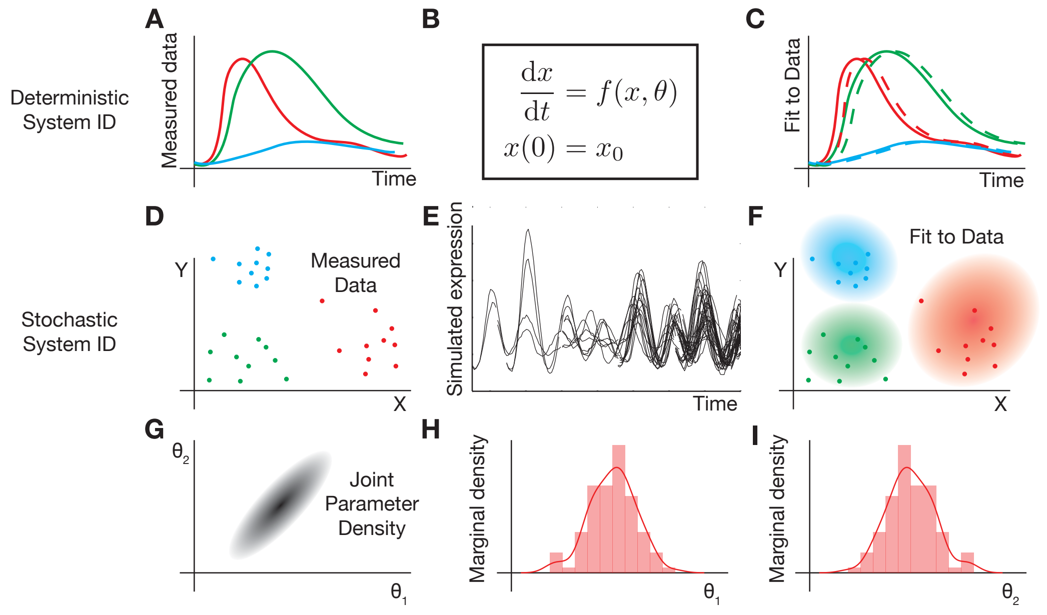


Figure 2.1: Bayesian system identification for deterministic and stochastic synthetic gene circuits. (a) Deterministic data is collected on a bulk culture over time. (b) A model using ODE's is used to model the synthetic gene circuit dynamics in a deterministic fashion. (c) The model fit is evaluated by matching simulated and measured trajectories. (d) Stochastic data is collected by measuring a distribution of outputs X and Y across a population of cells. (e) Stochastic simulations are used to compute the distribution of the output given a stochastic model. (f) The fit of the model to the data is evaluated by comparing output distributions across the population (g,h,i) The output of the procedure in both cases is a posterior density function over parameters. The joint parameter density function can show correlations between parameters, suggesting parameter degeneracy.

These measurement types and the associated output data are illustrated in Figure 2.2. Bulk data consist of measurements of an output for a culture of cells growing in a container, where the measured output can be thought of as the sum of the outputs for all the cells in the culture. Population snapshot measurements consist of measuring an output for a large number of individual cells in a population. However, the same cells cannot be followed over time, and at the next time point, a different set of cells will be measured. Population snapshot measurements include techniques such as flow cytometry and mRNA FISH (fluorescence in situ hybridization). Finally, with time series measurements, a single cell's output can be tracked over time using techniques such as time lapse microscopy.

It is important to note that each of these measurement types provides progressively more information than the previous type. If we ignore the time series relationships in time series data, then it can be considered as a set of population snapshots at different time points. If we take the mean or sum of a population snapshot output distribution, we can think of the resulting value as a bulk measurement.

Finally, the type of measurement collected can and should impact the choice of model for the system being studied. As an example, if the data collected is all bulk data, then using a stochastic model that accounts for noisy gene expression in single cells could be unnecessarily complex, and a deterministic model based on ODE's might be a better choice.

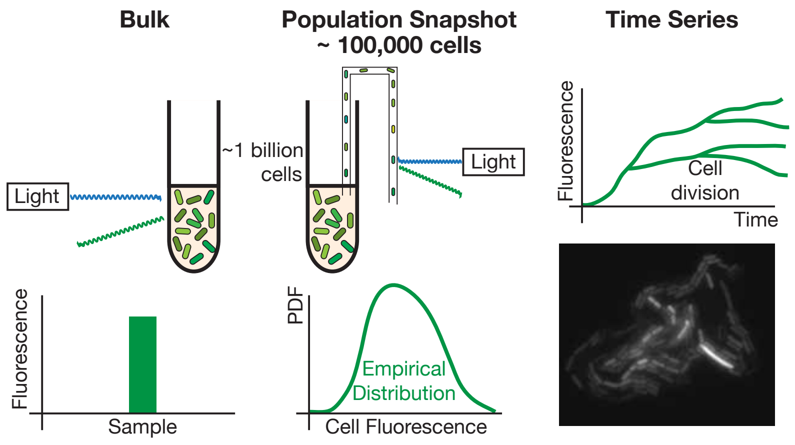
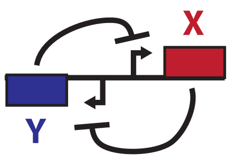


Figure 2.2: The different types of measurements for synthetic gene circuits. In bulk measurements, total fluorescence and cell population (by measuring optical density) can be measured for a culture over time using a plate reader. In flow cytometry, single cells pass through a detector that measures fluorescence for many individual cells. In time lapse microscopy, fluorescence output can be tracked across a growing lineage of single cells.

2.2 Model Classes

Synthetic gene networks are typically modeled as systems of chemical reactions occurring inside a cell where the chemical species in the circuit, proteins and mRNA's, are produced and degraded with different reaction rates. The level of one species can affect the rate of production or degradation of another species. This type of interaction is what gives rise to circuit type behavior.

A



B

$$\dot{X} = L + \frac{\beta}{1 + \left(\frac{Y^n}{K^n}\right)} - \delta X$$

$$\dot{Y} = L + \frac{\beta}{1 + \left(\frac{X^n}{K^n}\right)} - \delta Y$$

C

Parameter	Value
L	0.2
β	5
K	100
n	2
δ	2

D

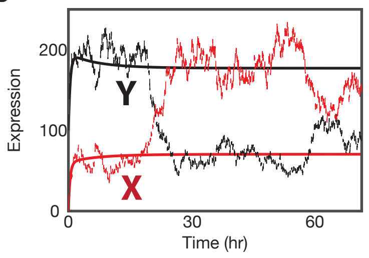


Figure 2.3: Modeling of a toggle switch synthetic gene circuit. (a) In a two species toggle switch, two genes X and Y repress each other's expression, which creates two steady states where either X is high and Y is low or vice versa. (b,c) Deterministic ODE's and parameters for a toggle switch model. Ignoring mRNA, X and Y are produced at a basal level with additional production repressed by the other gene along with first order dilution/degradation. (d) A deterministic (solid) and stochastic (dashed) simulation of the toggle switch. The deterministic simulation stays at a Y high/X low steady state while the stochastic simulation can switch due to noise.

For example, in Figure 2.3a, a simple synthetic gene circuit where two genes X and Y repress each other's production can give rise to a bistable toggle switch. In models of gene circuits, it is also common to ignore mRNA production and degradation as mRNA is an intermediary between DNA and protein that typically has fast dynamics and no function of its own. In the model of the toggle switch in Figure 2.3b, the mRNA dynamics are ignored, and the model simply has the two protein levels X and Y repressing each other's production.

The systems of chemical reactions can be modeled deterministically or stochastically. Deterministic models consist of a set of ODE's (see Figure 2.3b). In the stochastic case, each reaction is modeled as a discrete event that occurs with a certain propensity [24]. This results in a continuous time discrete state Markov chain whose probability evolution is described by the chemical master equation [92].

Simulation trajectories of the deterministic system can be generated using a standard ODE solver. For stochastic models, sample paths of the system can be generated using a stochastic simulation algorithm [24]. Also, the time evolution of the full state distribution can be approximately computed by using techniques such as moment closure [58] or finite state projection [64]. In moment closure, a system of ODE's approximately describes the time evolution of the moments of the state distribution. In moment closure, the state space is truncated, so that the time evolution of the state distribution can be approximated by solving a high dimensional linear system of ODE's.

These methods all describe ways to model synthetic gene circuits as chemical reactions inside a single cell. However, some synthetic gene circuits operate at a multicellular level, and thus, cell growth and death must be modeled. In a deterministic setting, cell growth is typically modeled as logistic growth with a saturating carrying capacity that depends on the type of cells and their environment. In the stochastic case, it is important to note that when a cell divides, its contents must be partitioned between its daughter cells. The randomness of this partitioning can introduce further noise into gene circuit dynamics.

When matching synthetic gene circuit models to experimental data, it is necessary to choose a model appropriate for the type of data being collected. For example, if bulk population average data is being collected, then a deterministic model might be more appropriate than a stochastic model. It can also be necessary to iterate on the reactions themselves to get the best fit to the data. A number of software packages exist that assist in modeling, simulation, and parameter inference of synthetic gene circuits [1,55,87,97].

A deterministic and a stochastic simulation of the toggle switch circuit from Figure 2.3a are presented in Figure 2.3d. The deterministic simulation stays at a low X and high Y steady state, while the stochastic simulation can randomly switch to the opposite state due to noise.

2.3 Identifiability of Synthetic Gene Circuit Models

Theoretical and computational considerations of the identifiability of synthetic gene circuits tend to focus on whether circuit parameters are identifiable for stochastic models given single cell population snapshot data. This is because these types of data are not typically collected in other engineering disciplines. However, even for bulk data and deterministic models, there are specific issues with system identification for synthetic gene circuits.

As the identifiability problem is difficult to assess, theoretical results tend to focus on linear systems where the theory is tractable. In [98], the observability problem for linear systems in which the linear system evolves a distribution over time was considered. Within this framework, the linear system represents the dynamics of a synthetic gene circuit, and the distribution at each time represents the distribution of the state of the circuit across a large population of cells. The output is then the distribution of an output for all time. The observability problem is to reconstruct the initial distribution of the system given the output distribution at every time. This distributional observability problem was shown in [98] to be strictly harder than the standard observability problem of finding a single initial state from a single output trajectory. Furthermore, it was proven that a sufficient condition for reconstructing the initial state distribution is to have an output dimension of at least $n-1$, where n is the state dimension. Finally, the problem of structural identifiability of linear systems driven by white noise is considered in Chapter 3 of this thesis. In this system, the entire steady state covariance of the system can be measured over time, but only certain subsets of state can be measured in time series. It is assumed that even though the system itself is stochastic, measurements are perfect and the steady covariance and output correlations can be measured exactly. Results from [88] showed that a combination of the steady state distribution and output correlations can be sufficient to determine the dynamics of the system exactly. This work is relevant to biological circuits because often many species can be measured simultaneously in steady state population snapshot measurements, but only a few species can be measured simultaneously in time series measurements.

Computational approaches to the identifiability problem rely on a variety of simulated data sets for a stochastic genetic circuit in order to perform system identification. In [65], the problem of identifying the parameters of a stochastic model of a genetic toggle switch given bulk data on the circuit, marginal distributions of the species, or the full joint distribution of the chemical species in the system is addressed. The results show that only having the full joint distribution is sufficient to recover the true parameters of the model without any parameter degeneracy.

2.4 Identification of Deterministic Biological Circuits

Techniques for identification of deterministic models of biological circuits are often quite similar to the techniques used for general identification of nonlinear systems. However, for synthetic gene circuits, data can typically be collected only for a few outputs at limited time resolution. Limited time resolution makes it hard to filter the time derivative of the output, which makes it difficult to apply a lot of traditional system identification methods that rely on the derivative of the measured output. Furthermore, most biological models contain many species, reactions, and parameters, and because only a small number of outputs can be measured, there are often many sets of parameters that can fit the measured data equally well. This effect is known as parameter degeneracy and can make it difficult to resolve the best parameters for the modeled system. Many techniques for system identification for synthetic gene circuits focus on alleviating the issue of parameter degeneracy.

For example, the extended Kalman filter is often used for parameter estimation, where the state is augmented with the parameters. However, [48] demonstrated that in a biological setting, matching the noise characteristics of the filter in addition to fitting the trajectory to measured data can help resolve parameter degeneracy. In [31, 34], relaxation type procedures are used to consider parameters sets that are consistent with the measured data, resulting in a set of regions in parameter space that fit the data within some error tolerance. Additionally, standard techniques based on Markov chain Monte Carlo (MCMC) can also be used [90]. Another approach to system identification for nonlinear systems is to build a dictionary matrix based on candidate functions and then use ℓ_1 regularization to select only a subset of the candidate functions to model the dynamics [70]. However, this approach requires full state measurements, which limits its applicability to synthetic gene circuits.

2.5 Identification for Stochastic Biological Circuits from Population Snapshots

This chapter categorizes the identification methods for stochastic biological circuits from population snapshot data into three different types: methods based on optimization and the finite state projection, methods based on moment closure, and Bayesian methods based on stochastic simulations of the system and MCMC.

In the optimization-based methods, the finite state projection method [64] is used to create a large linear system whose state is approximately the distribution of the chemical species in the system. The state distribution time trajectory can be calculated by solving the linear system, and the likelihood function can be computed from the observed data using a multinomial likelihood function. This method has been used to discriminate between different models of yeast osmotic stress response [68]. In [65] and [66], the same method is applied, but instead of a multinomial likelihood, a ℓ_1 norm penalty on the difference between the observed and predicted distribution is used. These types of methods work well for systems with small state spaces because the full state distribution can be solved. However, for systems with many species or high molecular counts, the finite state projection yields increasingly huge linear systems which become prohibitively expensive to repeatedly evaluate. Note that while [64–66, 68] all use the finite state projection to do an optimization based fit of the parameters in their models, the same likelihood function could be used in a Bayesian inference scheme as well.

Moment closure methods are one way to get around the curse of dimensionality in modeling stochastic gene circuits. With moment closure methods, the moments of the system can be modeled over time using a system of ODE's and these simulated moments can be compared with the experimentally measured moments for different parameter sets.

These methods are known to struggle in cases where the distributions are bimodal, but in some cases using a moment-based inference method can successfully predict an observed bimodal experimental data set [96]. In other cases, moment based methods can be used to characterize the causal relationships between different genes [51]. In [78], moment based inference in conjunction with an optimal experiment design technique based on maximizing the expected Fisher information is used to characterize a light inducible synthetic gene circuit. The limitation of moment closure methods is that the equations for the Nth order moments usually depend on higher order moments, and so to avoid having an infinite dimensional system of ODE's, the higher order moments have to be assumed to have some known form. In some case, the higher order moments can be computed from the lower order moments if the distribution is of a known type such as a normal or log-normal distribution [82]. In other cases, the third order cumulants are assumed to be zero [96]. A data-based identification procedure based on moment closure has also been developed, in which the reaction rate parameters are identified from the mean moment equations and the covariance values are plugged in from experimentally measured data [10]. In this way, no prior assumption on the distributions of the chemical species are required. Finally, semidefinite programming can be used to compute upper and lower bounds on the moments of a stochastic differential equation [47]. This approach is more computationally expensive than simple moment closure but provides a theoretical guarantee that the moment of interest lies within a specific interval.

Finally, the third class of methods for identifying circuits from population snapshot data are methods based on repeated stochastic simulations of the system. Some methods are based on approximate Bayesian computation [74], where the data is simulated for many different parameter sets and parameter sets that produce results in agreement with experimental data are kept. The INSIGHT method developed in [49] uses concentration inequalities to bound the deviation between the experimentally observed cumulative distribution function (CDF) of a certain output and the simulated CDF for a given parameter set. This deviation decreases as the sample size increases. With this technique, the method is able to quickly reject bad parameter sets after a very small number of simulations, vastly improving computation speeds.

While stochasticity is one way to explain the variation in circuit output across a population of cells, another possible explanation is heterogeneity between cells.

While variation in circuit output between cells can be explained by modeling chemical reactions as occurring stochastically, such variation can also be explained using deterministic models in which the variation is explained by differences in model parameters and initial conditions. Not every cell has exactly the same environment, and if different cells have different parameters and initial conditions, then the output will follow a distribution even if the circuit dynamics within each cell are deterministic. In [32] and [30], there is both the assumption that the circuit dynamics are deterministic, and that parameters for the cells are sampled from an underlying distribution or that there is a discrete number of cell types. This leads to a Bayesian inference procedure that can recover the parameters for each cell type, or the parameter density for a heterogenous population of cells.

2.6 Identification for Stochastic Circuits from Time Series Data

Identification of stochastic biological circuits from single cell time series data is typically done using methods based on Approximate Bayesian Computation or particle filtering. By selecting parameter sets and simulating a data set for each set of parameters, it is possible to perform Bayesian inference using MCMC without having to explicitly compute likelihoods [56]. This method is used in [90] to perform system identification on a simulated data set generated by a stochastic model of a repressilator. This type of likelihood-free method does not work well when the data is of a large dimension, and a better method is the particle marginal Metropolis Hastings method, where a particle filter is used to approximately compute the likelihood of the data given a set of parameters [26]. In [26], methods for speeding up the inference by using diffusion approximations to the stochastic simulation procedure are proposed.

Heterogeneity between cells can also be an issue for system identification. In [97], the heterogeneity between cells is modeled by assuming each cell has extrinsic factors and rate parameters distributed according to a gamma distribution. This choice of distribution allows for integrating out the extrinisic factors and parameter uncertainty while performing Bayesian inference. This approach was termed ‘dynamic prior propagation’, and demonstrated on experimental data obtained from an inducible synthetic circuit in yeast.

Chapter 3

Linear System Identifiability from Distributional and Time Series Data

3.1 Introduction

In this chapter, we consider the combination of distributional and time series measurements in system identification [88]. Distributional measurements consist of measuring the distribution of an output across an ensemble of systems at one given time. Time series measurements consist of measuring the output of one system over time and are standard in control theory. The central idea of this chapter is that it can be advantageous for system identifiability to combine distributional and time series measurements.

In synthetic and systems biology, we are often interested in experimentally probing the dynamics of biomolecular circuits in single cells. Distributional measurements in biology such as mRNA FISH (fluorescence in situ hybridization) [53, 75] and flow cytometry allow for quantifying mRNA or protein abundance for many genes across thousands of single cells. This results in output histograms as shown in Figure 3.1. Time series measurements such as time lapse fluorescence microscopy [95] allow us to measure a single cell's fluorescence over time as illustrated in Figure 3.1. However, time lapse microscopy measurements suffer from low output dimensionality and small sample sizes when compared to distributional measurements.

Therefore, our central idea is that it can be advantageous to combine the dynamic information from time series measurements with the high dimensional information from distributional measurements when investigating system dynamics.

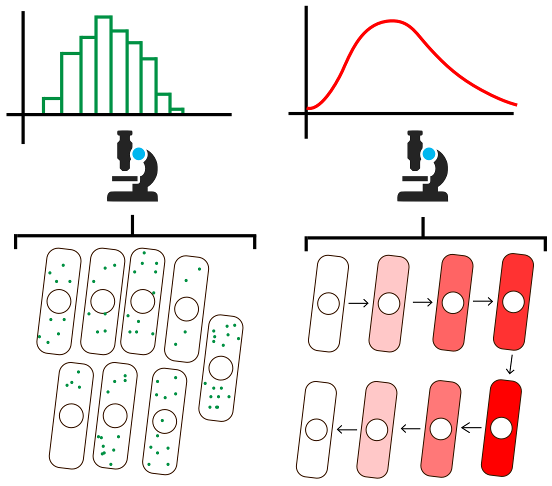


Figure 3.1: Population snapshot measurements generate output histograms (left) while time series measurements generate output trajectories (right).

It is commonly known that single cells exhibit heterogeneous behavior. [18, 20]. Common biological models for stochastic single cell behavior include stochastic linear systems [71], dynamic Bayesian networks [41, 79], and stochastic chemical reaction networks [24]. While stochastic chemical reaction models are more physically relevant, for theoretical tractability, we restrict our attention in this work to linear system models, which might still accurately model equilibrium fluctuations in a stationary stochastic process [71].

Linear system identification from time series data is a very well studied problem [44]. System identification from distributional data is less common but typically arises in inference of single cell dynamics from flow cytometry measurements [49, 65, 96]. Distributional measurements in the form of sample covariances have also been used in machine learning to find sparse graphical models [4, 15]. The combination of distributional and time series data for fully observable discrete time stochastic linear systems was considered computationally in [38]. The theory of dynamical structure functions has also been applied to identifiability of stochastic linear systems [12] as well as identifiability when different subsets of the state can be measured [94].

Importantly, Anderson [2] and Glover [25] considered system identifiability for stationary stochastic linear systems from output correlation measurements. In this work, we rely heavily on Anderson's and Glover's results. Our contribution is a consideration of stochastic linear system identifiability at stationarity given a combination of output correlation measurements in addition to the full steady state distribution. We term this the sensor fusion problem, because we are fusing distributional measurements with time series measurements for identification. Like Anderson and Glover, we assume that we have perfect measurements.

The structure of this chapter is as follows. In Section 3.2, we cover some preliminaries on linear systems driven by noise and set up the sensor fusion problem we would like to solve. In Section 3.3, we consider the case where we have only distributional measurements available. In Section 3.4, we introduce a sufficient condition and an effective necessary condition for system identifiability in the sensor fusion case when distributional and time series measurements are available. In Section 3.5, we consider a detailed simple example, a more complicated toy example with decaying oscillations, and a biologically inspired example. We discuss future work and conclude in Section 3.6.

3.2 Problem Setup

3.2.1 Preliminaries

In this chapter, we are concerned with linear systems driven by noise, and we assume that the systems have reached stationarity. We use the continuous time system description given in (3.1).

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bw(t) \\ y(t) &= Cx(t) \\ \mathbf{E}[w(t)w(t+\tau)^T] &\sim I\delta(\tau), \end{aligned} \tag{3.1}$$

where $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times m}$, and $C \in \mathbf{R}^{p\times n}$ are real matrices. The state $x \in \mathbf{R}^n$ corresponds to physically relevant quantities, and $w(t)$ is a unit white noise process in m dimensions, where $m \le n$. The number of outputs measured is p, which is smaller than n. Furthermore, we assume that each row of C is a canonical unit vector, so that the output is a subset of the state. We refer to A, BB^T, and C as the dynamics, noise, and output matrix respectively.

The transfer function for this linear system is given by $G(s) = C(sI - A)^{-1}B$, and (A, B, C) is a state space realization of $G(s)$. A minimal state space realization of $G(s)$ is one that is both observable and controllable.

We use G^* to denote the conjugate transpose of a complex matrix G, and we use A^T to denote the transpose of a real matrix A. We also write $P \succ 0$ or $P \succeq 0$ to mean that a matrix P is positive definite or positive semidefinite respectively.

Finally, general linear systems can have a feed through term D directly from input to output so that $y(t) = Cx(t) + Dw(t)$. In this chapter, we assume that $D = 0$, and we will generally give state space realizations with only three matrices (A, B, C). However, in some cases we will give state space realizations with four matrices. In this case, the final matrix is the feed through matrix D.

We then make the following assumption for the remainder of the chapter.

Assumption 1. *Given the linear system* (3.1)*, we assume that A is Hurwitz stable and that (A, B) is controllable.*

For a review of linear control theory, see [99].

3.2.2 Distributional Measurements for Linear Systems

The following standard result, which can be found in [99], characterizes the stationary state distribution of a linear system driven by white noise.

Proposition 1. *Given Assumption 1, the stationary state distribution of the linear system* (3.1) *is a multivariate Gaussian distribution with zero mean and covariance P, where P is positive definite and is the unique solution to* (3.2)*.*

$$AP + PA^T + BB^T = 0 \tag{3.2}$$

Proposition 1 allows us to assume that the distribution we observe is Gaussian. Since the form of a Gaussian distribution is fully determined by its mean and covariance and the mean is zero, we can get full knowledge of the steady state distribution from the covariance alone. Assumption 1 means that (A, B) is controllable, which guarantees us a strictly positive definite covariance P. The Lyapunov equation (3.2) provides a constraint that A and B must satisfy in order to produce a given covariance P.

3.2.3 Time Series Measurements for Linear Systems

Time series measurements allow us to measure output trajectories. There are two main types of time series measurements depending on whether a controllable input to the system is available. We start with the more standard case.

3.2.3.1 Time Series Measurements with Controlled Input

If we consider $w(t)$ to be a controllable input in (3.1), then we can characterize the input output transfer function $G(s)$ of the linear system given in (3.1). The following standard result characterizes the space of state space realizations consistent with a transfer function. Again, see [99] for more information.

Proposition 2. *Given a transfer function $G(s)$ with a minimal realization given by (A, B, C), then $(\hat{A}, \hat{B}, \hat{C})$ is also a minimal realization of $G(s)$ if and only if*

$$\hat{A} = TAT^{-1} \quad \hat{B} = TB \quad \hat{C} = CT^{-1} \tag{3.3}$$

for an invertible transformation T.

Proposition 2 tells us that the state space matrices can only be resolved up to a change of coordinates. This only holds if the system is of minimal order.

3.2.3.2 Output Correlation Measurements

When the input is an unobservable noise as in (3.1), we can only measure the system output, so we cannot expect to recover the input-output transfer function of the system. However, we can observe correlations in the output, or equivalently the output spectral density.

Given the linear system (3.1), we can measure the time correlations in the output y. Since the process is at stationarity, this corresponds to estimating the correlation function

$R(\tau) = \mathbf{E}[y(t+\tau)y(t)^T]$. In the frequency domain, the spectral density is given by

$$\Phi(s) = G(s)G^*(s),$$

where

$$G(s) = C\,(sI - A)^{-1}\,B$$

is the input-output transfer function of the system.

The problem of finding state space matrices that generate a specified spectral density has been studied extensively by Anderson [2] and Glover [25]. We present some of their results here.

We first define the notion of a globally minimal system as done in [2] and [25].

Definition 1. *A minimal state space system (A, B, C) is* globally minimal *with respect to a spectral density function $\Phi(s)$ if $G(s) = C(sI - A)^{-1}B$ satisfies $\Phi(s) = G(s)G^*(s)$ and A has the smallest dimension possible for any solution to $\Phi(s) = G(s)G^*(s)$.*

Specifically, a minimal system can sometimes produce an output spectral density that can be reproduced by another minimal system of lower order. In this case, the original system is minimal but not globally minimal. This scenario occurs when the original transfer function contains an *all pass* component, which is a term like $\frac{s-a}{s+a}$ that cancels out when forming the spectral density. We make the following assumption with respect to output correlation measurements.

Assumption 2. *Given the linear system* (3.1)*, we assume that we know the system order n a priori, that A is Hurwitz stable, and that (A, B, C) is a globally minimal realization for the transfer function $G(s) = C(sI - A)^{-1}B$ for any choice of C that we use.*

Assumption 2 is strictly stronger than Assumption 1. Both loss of controllability and pole zero cancellation are unlikely occurrences within the space of all linear systems, so our hope is that these assumptions are not too restrictive. The following lemma from [2] characterizes the set of globally minimal systems consistent with a spectral density.

Lemma 1. *Given an output spectral density of $\Phi(s)$, let $\Phi(s) = Z(s) + Z^*(s)$, where $Z(s)$*

is positive real. Then, suppose (A, G, C, J) is a minimal realization for $Z(s)$. Consider the following equation (3.4)*.*

$$\begin{bmatrix} AP+PA^T & PC^T-G \\ CP-G^T & -J-J^T \end{bmatrix} = -\begin{bmatrix} B \\ D \end{bmatrix} \begin{bmatrix} B^T & D^T \end{bmatrix} \tag{3.4}$$

Then all globally minimal realizations of $G(s)$ such that $\Phi(s) = G(s)G^*(s)$ have a realization given by (A, B, C, D), where B and D satisfy (3.4) *together with $P \succ 0$. Also, if B and D together with $P \succ 0$ satisfy* (3.4)*, then $G(s) = C(sI - A)^{-1} B + D$ is a globally minimal solution of $G(s)G^*(s) = \Phi(s)$.*

Lemma 1 characterizes all globally minimal state space realizations consistent with the output spectral density. In addition, the minimal dimension of the positive real transfer function matrix $Z(s)$ sets the globally minimal dimension for the system. The following result from [25] describes the relationship between multiple globally minimal realizations.

Lemma 2. *If (A_1, B_1, C_1, D_1) is a globally minimal realization for $\Phi(s)$, then (A_2, B_2, C_2, D_2) is also a globally minimal realization if and only if there is an invertible T and symmetric Q such that*

$$\begin{aligned} A_1 &= TA_2T^{-1} \\ C_1 &= C_2T^{-1} \\ QA_1^T + A_1Q &= -B_1B_1^T + TB_2B_2^TT^T \\ QC_1^T &= -B_1D_1^T + TB_2D_2^T \\ D_1D_1^T &= D_2D_2^T. \end{aligned} \tag{3.5}$$

Lemma 2 gives us an algebraic condition that relates globally minimal system realizations to each other. It is clear by letting $Q = 0$ that a change of coordinates is sufficient to satisfy the conditions of Lemma 2, meaning that the space of consistent systems is at least as large as in the controlled input case. This makes it mathematically clear why a controlled input is superior for identification.

3.2.4 Sensor Fusion Problem

We then set up the identifiability problem when both distributional and time series measurements are available. We assume we have a linear system of a known minimal order n that may be parametrized. However, we assume no parametrization for our results. We are interested in recovering the dynamic matrix A and the noise matrix BB^T.

We know the matrix C, because we already know which system states we are measuring. This corresponds to measuring a certain protein concentration in a cell or the value of a node in a network. We also might measure multiple subsets of the state in separate experiments using different matrices C_1, C_2, et cetera. We assume that all measurements are perfect measurements with zero error and that measurements can be collected for an infinite period of time. Therefore, for each choice of C, we assume we can exactly estimate the associated output correlation function. We also assume that we can exactly estimate the stationary state covariance P.

With this information, we would like to assess system identifiability both using distributional measurements alone as well as using a combination of distributional and time series measurements.

3.3 Identifiability Using Covariance Measurements Only

In this section, we consider the set of dynamic matrices A that are consistent with a measured covariance matrix P. We assume in this section that the noise matrix BB^T is known as well. The following propositions are straightforward linear algebra results.

Proposition 3. *Define the linear transformation $L_P(A) = AP + PA^T$. The transformation L_P takes $n \times n$ real matrices to $n \times n$ symmetric matrices. Then, consider the linear system* (3.1) *with B given. Then, the linear system* (3.1) *satisfies Assumption 1 and has a steady state covariance of $P \succ 0$ if and only if $A = -\frac{1}{2}BB^T P^{-1} + N$ with $N \in \ker L_P$ and A is Hurwitz or equivalently (A, B) is controllable.*

Proof. Since L_P is a linear transformation that takes $n \times n$ matrices to $n \times n$ symmetric matrices, we can write down all solutions to the continuous time Lyapunov equation (3.2) by using the particular solution $-\frac{1}{2}BB^TP^{-1}$ and adding a term from the kernel. However, this does not guarantee A must be Hurwitz, so we add that condition in separately. Since $P \succ 0$, the controllability of (A,B) is also sufficient to imply that A is Hurwitz [99]. □

The space where A is not Hurwitz but rather marginally stable is generally a set of measure zero within the affine subspace of $\mathbf{R}^{n\times n}$ given by $-\frac{1}{2}BB^TP^{-1}+\ker L_P$. Furthermore, it is easy to see that L_P is surjective onto the space of symmetric matrices and therefore, by rank-nullity theorem, the dimension of $\ker L_P$ is $\frac{n(n-1)}{2}$.

This result is quite different in discrete time, and we briefly present it here for completeness.

Proposition 4. *Consider the discrete time linear system $x_{k+1} = Ax_k + Bw_k$, where w_k is a unit Normal random variable of appropriate dimension. Also, fix B and P so that $P \succ 0$ and $P \succeq BB^T \succeq 0$. Then, the system is stable with steady state covariance P if and only if $A = (P - BB^T)^{1/2}UP^{-1/2}$, where $UU^T = I$ and A is Schur stable or equivalently (A,B) is controllable.*

Proof. The steady state covariance for discrete time systems satisfies the Lyapunov equation $APA^T - P + BB^T = 0$ [99]. In the forward direction, A must be Schur stable, and A must satisfy the Lyapunov equation. If A satisfies $APA^T - P + BB^T = 0$, then let $F = AP^{1/2}$ and let $G = (P - BB^T)^{1/2}$. Then $FF^T = GG^T$ and applying Lemma 3 in [76] gives that $F = GU$ must hold, which gives the expression for A. The matrix A must then be Schur stable by assumption and (A,B) must be controllable since A is Schur stable and $P \succ 0$ [99].

In the reverse direction, substitution reveals that A satisfies the discrete time Lyapunov equation. If (A,B) is controllable, then the fact that $P \succ 0$ implies A is Schur. Then, the fact that A is Schur in addition to the fact that A satisfies the Lyapunov equation means that the steady state covariance will be P. □

These propositions show that neglecting strict stability of A, we can resolve the A matrix up to an affine subspace in continuous time and and up to an orthogonal degree of freedom in discrete time. If we can control BB^T, then multiple covariance measurements for different values of BB^T in continuous time may resolve A exactly. In discrete time, the associated Lyapunov equation is quadratic and thus we can only ever resolve A up to a choice of sign.

As an example, consider the discrete time dynamics given by $x_{k+1} = Ax_k + w_k$, with

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 \end{bmatrix}.$$

These dynamics are that of a damped oscillation where the state moves from one entry to the next in a damped fashion. Since w_k has covariance I, we can compute a steady state covariance of $P = 4/3 I$. Applying Proposition 4 shows that then $A = 1/2 U$, where U is orthogonal. If we are interested in obtaining the structure of A, steady state covariance information does not help since A can be any orthogonal matrix. Furthermore, even if we know a priori that A might be sparse, A can still be any permutation matrix scaled by $1/2$, so it still does not help recover the structure of A.

3.4 Sensor Fusion

As stationary distribution information alone is inadequate, we now consider the combination of distribution and time series measurements. We first give a simple sufficient condition for identifiability of the dynamic matrix from noise driven measurements. Recall that the output matrix C allows us to measure a small number of elements of the state.

Proposition 5. *Assume the conditions of Assumption 1. Given a collection of time series measurements with different C matrices selected as $C_1, C_2, C_3, \ldots$, if each pair of states is measured together at least once, then the dynamic matrix can be identified exactly.*

Proof. The proof follows from the autocovariance function. If all pairs of states are observable together, we can reconstruct the system's autocovariance function $R(\tau) = \mathbf{E}[x(t+\tau)x(t)^T]$. We can calculate the derivative of the autocovariance function at zero as $\bar{R} =$

$\frac{dR}{d\tau}\big|_{\tau=0} = AP$. We also know that $R(0)$ is the steady state covariance P. We can recover $A = \bar{R}R(0)^{-1}$. □

Proposition 5 gives us a sufficient but highly conservative condition for the use of output correlation measurements to infer the system dynamics. If we have a system with 10 states and can simultaneously measure 2 outputs at a time, then we would need to conduct $\binom{10}{2}$ or 45 experiments to guarantee that we could resolve the system exactly.

In the sequel, we develop an effective necessary condition that requires far fewer experiments. To do this, first of all we extend Lemma 1 to the sensor fusion setting. We start with the following lemma.

Lemma 3. *Given a stable transfer function $G(s)$ with strictly proper realization (A,B,C), the positive real transfer function $Z(s)$ as defined in Lemma 1 must be strictly proper.*

Proof. From [25], we know that $Z(s)$ is the Laplace transform of the autocovariance function $R_y(\tau)$ of the output. We know that $R_y(0) = CPC^T$ is finite and that $R_y(\tau)$ exponentially decays to zero since the original system is strictly proper. Therefore, the Laplace transform $Z(s)$ exists for all non-negative s. Then, by applying the initial value theorem, we know that $Z(s)$ goes to zero as s goes to infinity so $Z(s)$ must be strictly proper. □

We can then prove the following theorem.

Theorem 1. *Assume the conditions of Assumption 2. Given a steady state covariance P and an output spectral density of $\Phi(s)$, let $\Phi(s) = Z(s) + Z^*(s)$, where $Z(s)$ is positive real. Then, apply Lemma 3 and suppose $(\hat{A}, \hat{G}, \hat{C})$ is a strictly proper minimal realization for $Z(s)$. Then consider the following equation* (3.6)*.*

$$\begin{aligned} AP + PA^T + BB^T = 0 \qquad PC^T = T\hat{G} \\ A = T\hat{A}T^{-1} \qquad CT = \hat{C} \end{aligned} \tag{3.6}$$

Then (A,B,C) is a globally minimal system with output spectral density $\Phi(s)$ and steady state covariance P if and only if *(A,B,C) satisfies* (3.6) *for some invertible T.*

Proof. The proof is a straightforward extension of Lemma 1, where we can set $J = D = 0$. In the reverse direction, if (3.6) holds, then the top left expression gives a covariance of P

immediately. Then, define $G = T\hat{G}$ and note that (A, G, C) is a coordinate transformation of $(\hat{A}, \hat{G}, \hat{C})$, so (A, G, C) is a valid realization for $Z(s)$. The reverse direction of Lemma 1 guarantees the desired output spectral density. In the forward direction, by global minimality we know that $\hat{A}$ from our realization for $Z(s)$ must be a coordinate transformation T away from the true A. Then, a different realization for $Z(s)$ with the true A is given by (A, G, C) with $G = T\hat{G}$ and $C = \hat{C}T^{-1}$. The forward direction of Lemma 1 implies that $PC^T = T\hat{G}$ and $AP + PA^T + BB^T = 0$ for some $P \succ 0$. However, that last equation is the covariance equation and so it must hold for the given covariance P. □

Theorem 1 fully characterizes the space of systems consistent with a combination of a steady state covariance and one output correlation. We consider the system identifiable if we can resolve A and BB^T uniquely from this information. We can search for a solution to the equations in Theorem 1 by using the coordinate transformation T and the noise matrix BB^T as variables. We can extend this result to the case where we measure the steady state covariance along with multiple different combinations of states as outputs. We first prove the following lemma.

Lemma 4. *Assume that $A \in \mathbf{R}^{n\times n}$ has no eigenvalues with multiplicity greater than one. It follows for invertible transformations T_1 and T_2 that $T_1AT_1^{-1} = T_2AT_2^{-1}$ if and only if $T_1 = T_2S\Lambda S^{-1}$, where S is a matrix whose columns are eigenvectors of A, and Λ is an invertible diagonal matrix. Λ must have real entries corresponding to real columns of S and complex conjugate entries corresponding to complex conjugate columns of S.*

Proof. Multiplying the equality by the appropriate quantities, we get $T_2^{-1}T_1A = AT_2^{-1}T_1$. This holds if and only if A and $T_2^{-1}T_1$ are simultaneously diagonalizable. Since A has no duplicate eigenvalues, the eigendirections are fixed, so we can write $T_1 = T_2S\Lambda S^{-1}$, where S contains the eigenvectors of A. The matrix Λ must have nonzero real entries corresponding to real columns of S and nonzero complex conjugate entries corresponding to complex conjugate columns of S. This is because T_1 and T_2 must be real. In the reverse direction, substitution shows $T_1AT_1^{-1} = T_2AT_2^{-1}$. □

Lemma 4 tells us that if two similarity transformations T_1 and T_2 both transform a matrix

A to the same final matrix, then given T_1, we have only n degrees of freedom when choosing T_2. This is because we have one real degree of freedom for each real eigenvector and one complex degree of freedom for each pair of complex eigenvectors. We can then formulate an effective necessary condition.

Let Assumption 2 hold and also assume that the dynamic matrix A has no duplicate eigenvalues. Suppose that we are given a steady state covariance $P \succ 0$ and a set of k output correlation measurements obtained with k different C matrices C_1 through C_k. Then, we would like to solve for the true A and BB^T matrices of the system.

Using the observed spectral densities $\Phi_i(s)$ where $1 \leq i \leq k$, we can compute each $Z_i(s)$ and find an $(\hat{A}_i, \hat{G}_i, \hat{C}_i)$ that realizes each $Z_i(s)$. By Assumption 2, all the $\hat{A}_i$ matrices will have the same dimension and be similar, so we can transform coordinates so that $\hat{A}_i = \hat{A}$ for all i. We then know that the true A matrix is given by $T_1\hat{A}T_1^{-1}$. The other T_i matrices must also transform $\hat{A}$ to A and are forced by Lemma 4 to satisfy the equation $T_i = T_1 S \Lambda_i S^{-1}$ for $i > 1$, where S contains the eigenvectors of $\hat{A}$. Then, we can write down a system of equations in terms of only T_1, BB^T, and Λ_i where $i > 1$.

Combining this with Theorem 1, we can rewrite the equations as

$$\begin{gathered} T_1\hat{A}T_1^{-1}P + P\left(T_1\hat{A}T_1^{-1}\right)^T + BB^T = 0 \\ PC_1^T = T_1\hat{G}_1 \quad C_1T_1 = \hat{C}_1 \\ PC_i^T = T_1 S\Lambda_i S^{-1}\hat{G}_i \quad C_i T_1 S \Lambda_i S^{-1} = \hat{C}_i \ \ \forall i > 1. \end{gathered} \tag{3.7}$$

This gives us a total of $\frac{n(n+1)}{2} + 2kpn$ equations. We have $\frac{n(n+1)}{2}$ variables for BB^T, n^2 variables for T_1, and an additional $(k-1)n$ variables for the Λ_i matrices. This gives a total of $\frac{n(n+1)}{2} + n^2 + (k-1)n$ variables. Then, in order to have more equations than variables, we get the following constraint on the number of measurements k.

$$k \geq \frac{n-1}{2p-1} \tag{3.8}$$

This inequality (3.8) is our effective necessary condition. This is not a true necessary condition because nonlinear equations in high dimensions can have unique solutions while having far fewer constraints than variables. For example, two hyper-spheres can be tangent at exactly one point. Also, if BB^T is not strictly positive definite, the positive semidefiniteness of BB^T may also help solve for A and BB^T. However, we typically expect to need at least as many constraints as variables to solve a system of equations.

There is no guarantee at all that the constraints generated from multiple experiments and equations will be independent, so more experiments might be required. In the case of ten nodes and two simultaneous outputs from before, condition (3.8) requires three measurements.

3.5 Examples

We now consider examples of sensor fusion.

3.5.1 Two Dimensional System

In this section, we consider sensor fusion for a two-dimensional system. This example leads to a negative result. We consider the system given by

$$\begin{aligned} A &= \begin{bmatrix} -1 & 1/2 \\ 1/2 & -1 \end{bmatrix} \\ BB^T &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ C &= \begin{bmatrix} 1 & 0 \end{bmatrix}. \end{aligned} \tag{3.9}$$

In this system, we also assume that we have prior knowledge that the noise enters the system only from the second state.

Using the Lyapunov equation (3.2), we compute a steady state covariance

$$P = \begin{bmatrix} \frac{1}{12} & \frac{1}{6} \\ \frac{1}{6} & \frac{7}{12} \end{bmatrix}.$$

The output spectral density of this system is given by $\Phi(s) = \frac{4}{16s^4 - 40s^2 + 9}$.

Then, we can apply Theorem 1 to compute the set of A and BB^T matrices consistent with the given P as well as the output spectral density of the system.

Using partial fraction decomposition, we can solve for $Z(s)$, and we find

$$Z(s) = \frac{\frac{1}{8}}{s + \frac{1}{2}} - \frac{\frac{1}{24}}{s + \frac{3}{2}}.$$

This verifies that the globally minimal system dimension is two, and we can write down a state-space realization for $Z(s)$ as $(\hat{A}, \hat{G}, \hat{C})$ below.

$$\hat{A} = \begin{bmatrix} -2 & -\frac{3}{4} \\ 1 & 0 \end{bmatrix} \quad \hat{G} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \hat{C} = \begin{bmatrix} \frac{1}{12} & \frac{1}{6} \end{bmatrix}$$

We can solve for all possible globally minimal systems consistent with this output correlation using Lemma 2. Using the true system as one globally minimal system, we solve for all other possible systems. First of all, the constraint $QC_1^T = 0$ implies that Q is all zeros except in the bottom right corner. Then,

$$A_1 Q + Q A_1^T + B_1 B_1^T = \begin{bmatrix} 0 & \frac{q}{2} \\ \frac{q}{2} & 1 - 2q \end{bmatrix},$$

where q is the bottom right entry of Q. Since this term must be positive semidefinite, then $q = 0$ must hold, and we see that $Q = 0$. Thus, the equations simplify down to $A_1 = TA_2T^{-1}$, $C_1 = C_2T^{-1}$, and $B_1B_1^T = TB_2B_2^TT^T$. We can solve these equations assuming T is a free two by two matrix and assuming that $B_2B_2^T$ is a symmetric matrix with all zeros and one positive entry in the lower right hand corner.

Then, solving the equations gives us the following expressions for the set of consistent A_2 and $B_2B_2^T$ matrices for this system.

$$A_2 = \begin{bmatrix} \frac{c}{2} - 1 & \frac{d}{2} \\ -\frac{c^2-1}{2d} & -\frac{c}{2} - 1 \end{bmatrix}$$

$$B_2 B_2^T = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{d^2} \end{bmatrix}$$

$$c, d \in \mathbf{R}$$

$$d \neq 0$$

Time series measurements along with knowledge of the noise structure do not provide much information about the structure of A in this case. Of course, the steady state distribution is also insufficient to determine A as shown in Proposition 3.

The next step is to consider sensor fusion. In the sensor fusion case, we solve the equations given in Theorem 1. These equations can again be solved by letting T be a free variable. In this case, the noise matrix BB^T can be found exactly. However, the A matrix has two possible solutions. The first solution is the correct A matrix which leads to the transfer function $G(s)$ above. The other solution is

$$A_{\text{wrong}} = \begin{bmatrix} 1 & -\frac{1}{2} \\ \frac{15}{2} & -3 \end{bmatrix},$$

which leads to a transfer function of $-G(s)$, which of course reproduces the same spectral density $\Phi(s)$ but also reproduces the same covariance at steady state. With sensor fusion, there are exactly two consistent A matrices.

Interestingly, this result seems to hold for different output choices. For example, we could measure the second state instead of the first state or the difference of the two states. They all produce either the true A matrix or the incorrect A_{wrong}. In this case, sensor fusion cannot exactly determine A, but it does provide us with two specific possible values of A instead of the one or two dimensional infinite space that we get from using distribution or time series measurements alone.

3.5.2 Cyclic Dynamics

In this example, we investigate sensor fusion for a three node decaying oscillation system.

We try a few different cases in terms of which nodes are measured in the system. We start with the three dimensional system given by

$$A = \begin{bmatrix} -1 & 0 & 1/2 \\ 1/2 & -1 & 0 \\ 0 & 1/2 & -1 \end{bmatrix}$$

with noise entering only from the third state, so that

$$BB^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The system is shown in Figure 3.2 with the first and second nodes measured together.

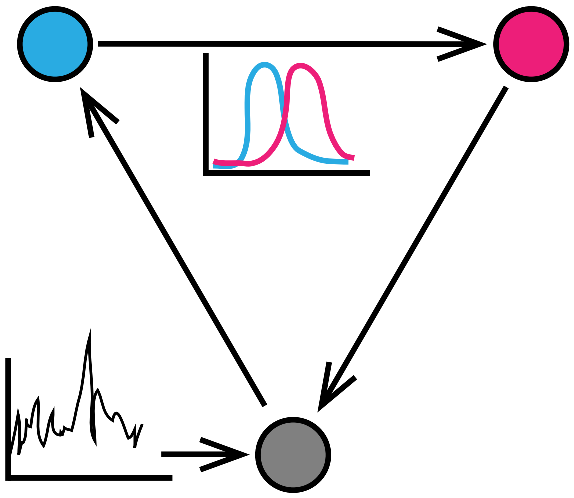


Figure 3.2: A three node ”oscillating” system with one noise input and two measured outputs.

The previous section just showed that measuring steady state covariance and a single output correlation is not sufficient to recover system dynamics for a two dimensional system. This limitation of single output measurements still holds for this system. We then consider the case of measuring two outputs simultaneously. This is the situation illustrated in Figure 3.2. Noise enters into state three while states one and two are measured simultaneously.

In this case, the algebra associated with Lemma 2 leads to another messy expression for the space of possible A matrices consistent with output correlations alone.

Incorporating the steady state information in this case again leads to two possible A matrices while recovering BB^T exactly. One of course is the correct A, while the other is given by

$$A^{12}_{\text{wrong}} = \begin{bmatrix} 3 & -\frac{31}{6} & -\frac{1}{2} \\ \frac{1}{2} & -1 & 0 \\ \frac{161}{6} & -\frac{251}{6} & -5 \end{bmatrix}.$$

If we measure the first and third states together, then we again get two possibilities for A, where the incorrect one is given by

$$A^{13}_{\text{wrong}} = \begin{bmatrix} -1 & 0 & \frac{1}{2} \\ -\frac{65}{62} & 1 & \frac{12}{31} \\ \frac{508}{31} & -\frac{63}{2} & -3 \end{bmatrix}.$$

From these results, it is clear that if we measure steady state covariance information along with output correlations for the first and second state together as well as output correlations for the first and third state together, we can resolve A exactly.

Interestingly, if we measure the output correlations for states two and three together, then that alone is sufficient when combined with the steady state covariance to resolve A exactly. That is, we can resolve A exactly without ever measuring state one in time series. Additionally, it is clear that identifiability is a question of which nodes we measure in addition to how many nodes we measure. This suggests a consideration of graph structure and identifiability in the same vein as well known graph theoretic analyses of controllability and observability [50].

3.5.3 Example - RNA Elongation and Degradation

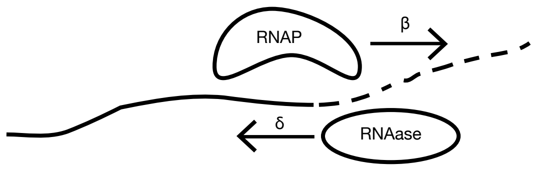


Figure 3.3: A transcription and degradation process for mRNA

In this section, we consider the biologically inspired example of a transcription degradation system at equilibrium. RNA Polymerase is an enzyme that transcribes and lengthens the RNA molecule polymer while RNAase is an enzyme that degrades and shortens the RNA. We illustrate this system in Figure 3.3 and model it using the dynamics in (3.10).

$$\begin{aligned}\dot{x} &= b + Ax + Bw \\ x, b, B &\in \mathbf{R}^n \\ A &\in \mathbf{R}^{n\times n}\end{aligned} \tag{3.10}$$

Here, the elements of x correspond to varying lengths of partial RNA where $x(1)$ is a nascent transcript and $x(n)$ is a fully completed transcript. The rate matrix

$$A = \begin{bmatrix} -(\delta_1+\beta_1) & \delta_2 & & & \\ \beta_1 & -(\delta_2+\beta_2) & \ddots & & \\ & \beta_2 & \ddots & \delta_{n-1} & \\ & & \ddots & -(\delta_{n-1}+\beta_{n-1}) & \delta_n \\ & & & \beta_{n-1} & -\delta_n \end{bmatrix}$$

is a tridiagonal matrix where the terms represent the local rates of elongation and degradation. Namely, the superdiagonal terms δ_k represent the local rates of degradation at different points along the transcript, the subdiagonal terms β_k represent the local rates of elongation at different points along the transcript, and the diagonal terms enforce conservation. Note that the δ_k's and β_k's must be positive. As constructed, A is diagonally dominant and thus Hurwitz.

We model the actual transcription and degradation processes as proceeding deterministically with noise being introduced only by initiation events. The bias term $b = \begin{bmatrix} b_1 & 0 & \cdots & 0 \end{bmatrix}^T$ with $b_1 > 0$ gives the mean amount of transcription initiation activity. The noise term $B = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T$ is the noise in the transcription initiation process.

We assume that we can measure the steady state distribution of this system, which we might do by looking at single cell RNA data across a population of cells. Because of the bias term, we would not only be able to measure the covariance at steady state, but we could also measure the mean x_{eq}, where $Ax_{eq} + b = 0$ must hold at steady state.

There are a total of $2n + 1$ total parameters to identify. The rate matrix A has $2n - 1$ parameters, and the noise and bias term have one parameter each. Using the steady state mean information alone gives only the equation $Ax_{eq} + b = 0$. This only provides n linear equations, so we cannot solve for all $2n + 1$ parameters.

However, using the steady state distribution gives us more information. First of all, it is clear from the structure of A and B that (A, B) will be almost certainly be controllable as the noise can propagate to every state and therefore, the steady state covariance P will be strictly positive definite and unique. Knowing the covariance provides us with another $\frac{n(n+1)}{2}$ equations through the continuous time Lyapunov equation (3.2).

However, since all the equations are linear, we can only determine the parameters up to multiplication by a positive scalar. Intuitively, this is because rescaling time does not affect steady state measurements. Therefore, without some additional information, we still cannot determine A exactly.

For a specific numerical example, we set

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

and we set $b = B = [1\ 0\ 0\ 0]^T$. In this example, steady state measurements constrain A to a one dimensional subspace of $\mathbf{R}^{n\times n}$. The set of dynamics consistent with the steady state mean and covariance is given by kA, where $k > 0$. Note that $k < 0$ is impossible because it would make A not Hurwitz. The set of consistent B matrices is given by kB.

Then, suppose we measure the output correlation with $C = [0\ 0\ 0\ 1]$, which is the state corresponding to fully completed RNA molecules. This results in a spectral density of

$$\Phi(s) = \frac{1}{s^8 - 19s^6 + 87s^4 - 70s^2 + 1}.$$

Since we know the true A and B matrices up to a scalar factor k, the set of spectral densities consistent with the distributional information is given by $\Phi\left(\frac{s}{k}\right)$. We see that $k = 1$ must hold, which means that sensor fusion is sufficient to resolve A, b, and B exactly for this example.

3.6 Conclusion and Future Work

In this chapter, we considered the idea of combining full state covariance information together with limited output correlations to improve identifiability of stationary linear stochastic systems driven by white noise. This problem was motivated by the biological setting, where often distributional measurements are high dimensional and time series measurements are low dimensional. We formulated conditions for identifiability using distributional measurements alone and also for using sensor fusion.

Future work includes developing a practical identifiability algorithm for sensor fusion that is robust to noise. This algorithm will complement the theoretical identifiability analysis developed in this chapter. We then hope to demonstrate both sets of techniques on data generated from either a simulated or experimental stochastic biological circuit to demonstrate the utility of the methods.

Chapter 4

Quantitative Modeling of Integrase Dynamics Using a Novel Python Toolbox for Parameter Inference

4.1 Introduction

Quantitative approaches to both synthetic and systems biology rely on modeling and simulation of biological circuits. Biological circuits are systems where different components, typically genes, can interact with each other via different types of molecular interactions including transcriptional activation and repression and sequestration. These interactions are typically modeled as a system of chemical reactions where each reaction has a certain reaction rate. The rate can be modeled using mass action kinetics [13], but many reactions in biological circuits have sigmoidal saturating reaction rates. These reactions are typically modeled using Hill functions [13] .

Given the reaction model for a biological circuit, there are many different ways to simulate the model. First of all, regardless of simulation framework, it is necessary to specify the values of the parameters in the model along with the initial levels of the model species. By treating the reaction rates as deterministic bulk rates and using ordinary differential equations (ODE's), one might perform a deterministic simulation of the system.

On the other hand, it also common to treat the reaction rates as propensities for discrete reactions to fire and to model the system as a set of stochastic chemical reactions [24]. Biological circuits can often be noisy [18, 20] especially at low molecular copy numbers, and a stochastic model is often necessary to capture the noise characteristics of a circuit.

Some other common features in biological circuit models are delay and cell growth and division. Processes like protein production are not instantaneous, and there is often a significant delay between when transcription of a gene is initiated and when a mature protein is produced. This type of delay can lead to nontrivial behavior such as oscillations [85], and thus it is often important to incorporate delay into the modeling framework. Additionally, delays can be both fixed and distributed in their duration. While adding a fixed delay to a biological circuit might destabilize the circuit and create oscillatory behavior, distributing that delay across multiple durations might maintain circuit stability [27].

Cell growth and division are also critical aspects of biological circuits that operate in single cells. Typically, a dilution term in the model accounts for cell growth. However, in stochastic models, modeling the continuous dilution process with a stochastic and discrete degradation reaction might not be accurate. Another source of noise is the partitioning of molecules between daughter cells at cell division [40]. In fact, it can be difficult to distinguish between noise in gene expression and noise from molecular partitioning [40]. Therefore, modeling cell growth as well as division and partitioning is important for investigating noise in gene expression across a lineage of cells.

In order to validate these types of models of biological circuits against actual data, it is often helpful to have a large amount of experimental data measured for the circuit. For deterministic models, it is often helpful to have data collected at many different operating conditions, while for stochastic models, it is also helpful to have large sample sizes. The increasing use of technologies for lab automation as well as high throughput measurement techniques involving liquid handling [63] and flow cytometry [79, 96] makes this data collection more easy to do.

Using experimental data, one can fit parameters for models of biological circuits and tune the models to better match qualitative trends in the data. In this workflow, it is often necessary to add or remove reactions from the model or to perform a different type of simulation. For example, one might decide that a circuit behaves too noisily for deterministic simulations and want to switch to a stochastic simulation framework. If delays are playing a significant role in the dynamics, one might want to incorporate previously unmodelled delays into the model.

The most attractive methods for performing parameter inference on these models are those that use Bayesian inference [26, 46], because these methods provide a full posterior distribution over the parameter space. This gives insight into the accuracy and identifiability of the model. Also, such an approach allows for an easy comparison between different model classes using the model evidence. The drawback of these approaches is that their implementation is computationally expensive and is based on repeated forward simulations of the model within the framework of Markov chain Monte Carlo (MCMC) [26]. Therefore, it is important to have the underlying simulations running as fast as possible in order to speed up computation time.

This chapter presents `bioscrape` (Bio-circuit Stochastic Single-cell Reaction Analysis and Parameter Estimation), which is a Python package for fast and flexible modeling and simulation of biological circuits. The bioscrape package uses Cython [5], an extension for Python that compiles code using a C compiler to vastly increase speed. This helps assuage the computational time issues that arise in parameter estimation. Additionally, `bioscrape` provides a flexible text-based framework for specifying models of biological circuits, as well as a set of simulators that can perform deterministic and stochastic simulations as well as incorporate delay and cell growth and division. If a researcher needs to change a circuit model, he or she can simply edit a few lines in a text file to make the change. If a researcher needs to simulate a circuit differently, he or she can simply select a different simulator to use.

Some popular software packages that do somewhat similar tasks to the bioscrape package are MATLAB's SimBiology toolbox [1] and Stochpy [55]. However, the bioscrape package runs simulations faster than either and also supports simulations of whole cell lineages as well as more general reaction rates.

The following sections contain an overview of the model specification language as well as some examples of simulations performed using the package. Additionally, Section 4.6 contains a demonstration of bioscrape to perform parameter estimation for integrase DNA recombination dynamics based on experimental data. More detailed documentation and the code for the examples as well the package itself are available online.[1]

4.2 A flexible modeling language for biological circuits

The first piece of the software package is a flexible text-based modeling framework. The framework uses XML to specify the reactions that make up the interactions of a model. Once the XML file for a model is loaded into the software package, it can then be simulated using whatever method is desired.

Another common XML based specification language for biological models is the Systems Biology Markup Language (SBML) [39]. The SBML language is very general and can describe a wide variety of biological systems, but it is also complex. Therefore, the bioscrape package instead uses a very simple modeling language that only specifies model reactions.

The XML language consists of a simple specification of reactions along with propen-

[1]The bioscrape package is available at https://github.com/ananswam/bioscrape.

sities and delays.

Code Block 4.1: Simple examples of XML reactions

```
<reaction text=”—” after=”—mRNA”>
    <propensity type=”massaction” k=”beta” species=”” />
        <delay type=”fixed” delay=”tx_delay” />
</reaction>

<reaction text=”mRNA—” after=”—”>
    <propensity type=”massaction” k=”delta_m” species=”mRNA” />
        <delay type=”none” />
</reaction>
```

Code block 4.1 contains an example specifying two reactions. The first reaction is a transcription reaction with delay. The `reaction` XML tag contains a `text` field and an `after` field, where the `text` field contains the part of the reaction that happens initially and the `after` field contains the part of the reaction that occurs after a delay. The products and reactants are separated from each other by the `--` characters, and multiple products and reactants can be included by separating them with plus signs.

The first reaction above is a delayed transcription reaction, as the only thing that happens in this reaction is that mRNA appears after a delay. The second reaction is a degradation reaction without delay as the only thing that happens is that mRNA disappears.

The `propensity` field specifies the reaction rate. In this case, both reactions have mass action propensities. For mass action propensities, the rate parameter is given by the attribute `k` and the `species` attribute specifies the species involved. In this case, mRNA is produced with a constitutive propensity of `beta`, so there are no species involved, and the rate is specified to be the parameter `beta`. For the second reaction, the degradation rate of mRNA is given by $\texttt{delta\_m} \times \texttt{mRNA}$. If more than a single species is involved in a mass action reaction rate, the species can separated by $*$ signs.

Finally, a delay must also be specified for each reaction. If there is no delay, the delay type can be set to `none` as in the second reaction. However, if there is a delay, the type of delay must be specified along with any associated parameters. The transcription reaction above has a fixed delay, and thus a parameter is required to specify the duration of the fixed delay. If the delay is distributed with a gamma distribution, then the shape and scale of the distribution must be specified.

Code Block 4.2 shows the initialization process for parameter values and initial species levels. Both the initial conditions and the parameter values can be modified from a Python script after the model has been loaded as well, so it is not necessary to edit the XML file and load the model every time a parameter or initial condition is changed.

Code Block 4.2: Initialization of parameters and species

```
<parameter name=”beta” value=”2.0” />
<parameter name=”delta_m” value=”0.2” />
<parameter name=”k_tl” value=”5.0” />
<parameter name=”delta_p” value=”0.05” />
<parameter name=”tx_delay” value=”10” />
<parameter name=”tl_k” value=”2” />
<parameter name=”tl_theta” value=”5” />

<species name=”mRNA” value=”0” />
<species name=”protein” value=”0” />
```

The reactions in Code block 4.1 and the parameters and initial species values in Code block 4.2 are both part of an XML file describing a simple model of gene expression with delayed transcription and translation and the usual linear degradation of mRNA and protein. This model will be simulated in many different ways in the next section, and the full model can be found in Appendix A.1.

4.3 Flexible Simulation of Biological Circuits

This section demonstrates the different types of simulations that can be performed with the bioscrape package using the model of delayed gene expression from Appendix A.1 as an example.

The package is capable of performing stochastic simulations with and without delay and with and without cell growth and division. However, deterministic simulations only work without delays or cell division.

As a first pass, the simple gene expression model can be simulated both deterministically and stochastically ignoring delays and cell division. Importantly, switching between a deterministic and stochastic simulation only requires a single line to be modified in the code. The simulation output is the mRNA and protein trajectories over time.

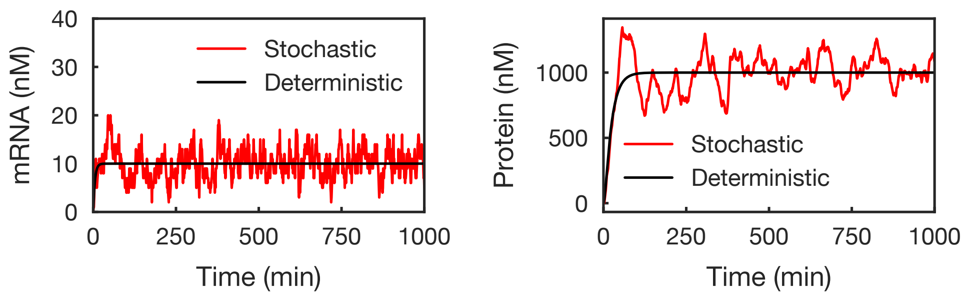


Figure 4.1: Deterministic and stochastic simulations of simple gene expression without delay.

Figure 4.1 shows the simulated deterministic and stochastic trajectories. As expected, the deterministic simulation smoothly goes to a steady state of 10 mRNA molecules and 1000 proteins, while the stochastic simulation bounces around the deterministic value. Because the mRNA production and degradation events in this model represent a standard birth death process, it can be analytically determined that the steady state distribution of mRNA should be a Poisson distribution with parameter $\lambda = 10$, and that the autocorrelation time for the mRNA trajectory should be $\exp(0.2t)$. Both of these analytical solutions can be compared to the empirical probability distribution and autocorrelation for mRNA. These can be computed using a longer simulation of 50,000 simulation minutes. The results are presented in Figure 4.2.

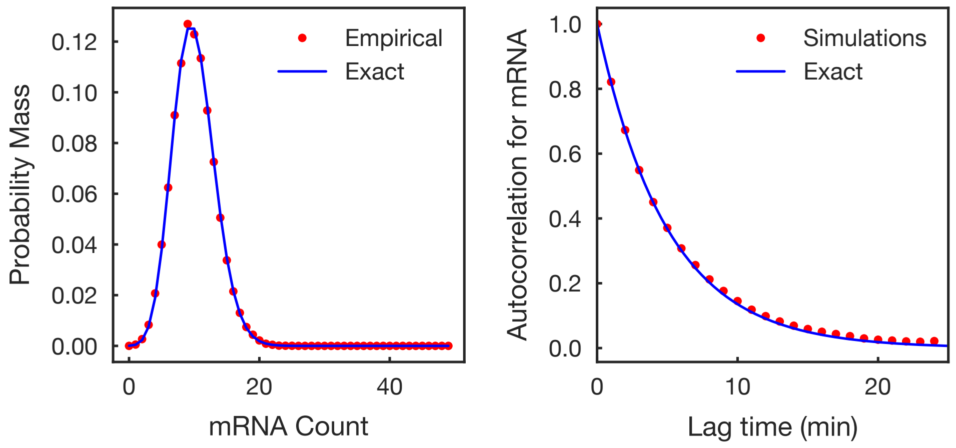


Figure 4.2: Empirical steady state mRNA distribution and autocorrelation times match analytical results.

The next step is to incorporate the delays specified in the model. The transcriptional delay in the model is a fixed delay of 10 minutes, while the translational delay is gamma distributed with a mean of 10 minutes. Again, switching from a simulation without delay to one that incorporates delay only requires changing a few lines of code. Only a single line of code is required to change the simulator, but as the initial conditions for a delayed simulation include a history of queued reactions, this initial condition must be constructed as well. The results from a delayed simulation of the model are presented in Figure 4.3. As expected, the mRNA turns on at 10 minutes, and the protein turns on at around 20 minutes. Because the delay is distributed, the protein actually starts increasing slightly before 20 minutes have elapsed.

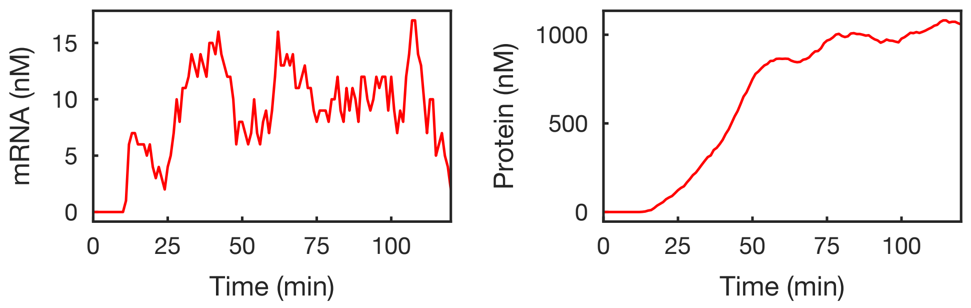


Figure 4.3: Delayed simulation shows an abrupt delay in mRNA production and a smooth delay in protein production.

Cell growth and division can also be incorporated into the simulation. The dynamics of cell growth and division are not specified in the model XML file, so they must be specified in the Python simulation script. The model must specify the rate of cell growth as well as the time and volume at which the cell divides. The model of cell growth and division used in the package consists of deterministic exponential growth and cell division upon reaching a certain volume threshold. A noise parameter allows for incorporating some stochasticity into the dynamics. The daughter cells in this model are also exactly half the size of the mother cell.

The method for partitioning molecules between daughter cells must also be specified in the simulation Python script. Currently, the model for molecular partitioning is one in which the molecules are partitioned binomially between the two daughter cells in a manner that is consistent with the daughter cell volumes. For example, if one daughter cell is twice as big as the other, then it receives about two thirds of the molecules.

The reaction rates in the model also become volume dependent when performing a volume-based simulation. For example, a unimolecular reaction does not depend on volume, but a bimolecular mass action reaction rate will scale as Volume^{-1} because the odds of two molecules finding each other decrease as the volume increases. Similarly, in Hill function rates, the transcription factor in the Hill function is replaced by its concentration when doing volume based simulations, since the concentration sets the equilibrium binding of the transcription factor to the DNA. These changes happen automatically when switching to a volume based simulation.

The output of a lineage simulation is a set of cell traces, where each trace contains a cell's volume and species trajectories over a single cell cycle. Each cell trace also may or may not have a parent cell trace as well as daughter cell traces. This data structure is similar to the one used in the Schnitzcells microscopy image analysis software [95].

A simulation of the simple gene expression model ignoring delays is provided in Figure 4.4 for a lineage of cells with a division time of 33 minutes. In this figure, the plotted quantity is protein concentration, which is molecules of protein per cell divided by the current cell volume. Here, the cell volume ranges between 1.0 and 2.0, and the volume unit can be thought of as a characteristic *E. coli* cell volume. A common assumption in *E. coli* models is that 1 molecule per cell corresponds to a concentration of 1 nM, and so 1 molecule per characteristic cell volume here is assumed to be 1 nM.

The initial single cell trajectory branches into more and more trajectories as the cell divides. At cell division, the concentration of protein in each daughter cell can change discontinuously. This occurs because of the noise in molecular partitioning. Because the proteins are high copy number, this noise is fairly small in this example.

The most comprehensive type of simulation, which can be performed using this model, is one that incorporates both cell lineages and delays. The simple gene expression model with delays can be simulated using the same model of cell growth and division (33 minute division time) as the one used to produce Figure 4.4. In this case, the transcription and translation delay are set to 16.5 minutes each, for a total delay in protein production of one 33 minute cell cycle. The results are shown in Figure 4.5.

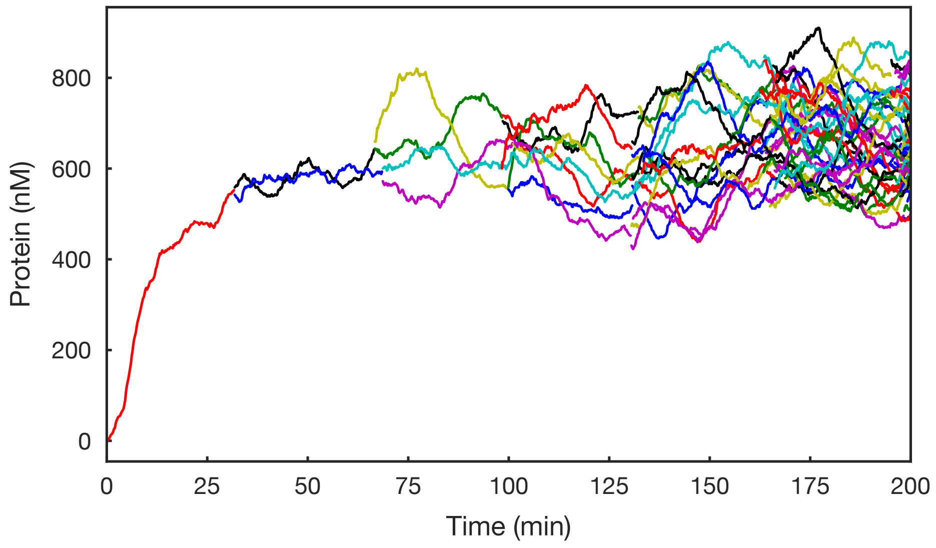


Figure 4.4: Simulation with cell growth and partitioning shows variation in gene expression across a lineage of cells.

While the results from Figure 4.5 look qualitatively similar to the results in Figure 4.4, it is important to note that the final level of steady state protein expression in the simulation with delay is only about 300 proteins versus the 600 protein steady state in the simulation without delay. The reason for this is that because of the delay, the amount of protein that is being produced at a given time is proportional to the number of cells that existed 33 minutes ago. Because the delay is one cell cycle, the amount of protein production is effectively half of what it would be without delay. This is consistent with the steady state protein expression with delay being about half as much as without delay.

This result provides insight for selection of fluorescent proteins for use in exponentially growing colonies of cells. It might be better to use fast maturing fluorescent proteins that are dimmer on average compared to bright fluorescent proteins that mature slowly, because the signal loss due to the long delay might outweigh the additional brightness per

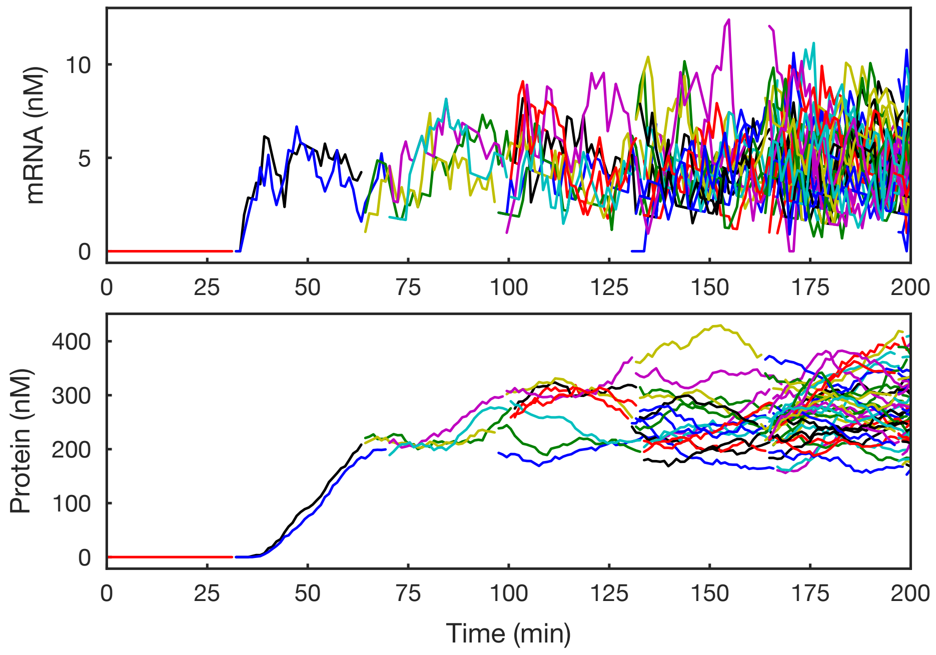


Figure 4.5: Simulation with cell growth and partitioning in addition to delay shows reduced steady state gene expression across a lineage of cells.

molecule. For example, a fluorescent protein that takes an extra cell cycle time to mature must be twice as bright in order to generate the same fluorescent signal at steady state.

4.4 Fast Simulation of Biological Circuits

In addition to the framework for flexible modeling and simulation of biological circuits described in the previous sections, the third critical aspect of a software package for quantitative analysis of biological circuits is speed. This package is written using Cython [5], a language extension for Python that creates compiled Python libraries. Some alternative methods for doing stochastic simulation are to use the SimBiology toolbox in MATLAB [1], write code in C from scratch, or to use a pure Python library such as StochPy [55]. In this section, the simulation speed of the bioscrape package is benchmarked against these other three common simulation options.

The benchmark test used for comparing the speed of these different simulators is to simulate the simple gene expression model from Appendix A.1. As MATLAB SimBiology does not support delayed reactions, the system was simulated ignoring delays for 100,000 minutes starting from an initial condition of zero. Additionally, both SimBiology and StochPy output each step of the stochastic simulation as opposed to outputting the system state at specific times. For the simulation conditions in this system, the number of steps taken in 100,000 minutes is always around ten million steps. Therefore, to make the comparison fair, the simulation in the bioscrape package is done with ten million desired time points in order to keep the output size the same in all cases. Finally, the C code is a pure C implementation of the simulation using the same fixed interval algorithm as the bioscrape Python package, so the C implementation is also run with ten million desired time points.

Table 4.1: A speed comparison between bioscrape and other common simulation platforms.

Software	**Benchmark time (s)**	**Speed-up**
SimBiology	5.8	8.3x
StochPy	190	270x
C	0.38	0.54x
bioscrape	0.70	-

The simulation times are available in Table 4.1. The table shows that the bioscrape package outperforms SimBiology by almost one order of magnitude, but it outperforms the pure Python StochPy package by a factor of 270. The C simulation is used to get an idea of the maximum speed possible. The bioscrape package is about twice as slow as custom pure C code. This is due to a choice to preserve code readability over absolutely maximizing speed.

4.5 Simulating Plasmid Replication and Gene Expression in a Cell Lineage

As a more complex demonstration of the capabilities of the bioscrape package, it can be used to model plasmid based transcription with cell growth and division. The plasmids in the model replicate and control their own copy number. Additionally, each plasmid constitutively expresses mRNA. This simulation can be used to get an idea of the variability involved in plasmid based transcription. In order to perform such a simulation, a simplified model for plasmid replication and copy number control is developed.

4.5.1 A Reduced Order Model for Plasmid Replication in Single Cells

The ColE1 plasmid regulates its own copy number by constitutively transcribing an RNA that inhibits the RNA primer for DNA replication from initiating a replication event [8]. Making a four simplifying assumptions enables the derivation of a simplified model of plasmid copy number regulation. First, it is assumed that the inhibitory RNA directly binds to the plasmid origin to inhibit replication. Second, it is assumed that that the replication rate is proportional to number of free plasmids, which do not have inhibitory RNA bound. Third, it assumed that the inhibitory RNA transcription and degradation dynamics are much faster than the plasmid replication dynamics. Fourth, the inhibitory RNA is assumed to be strongly transcribed and linearly degraded, so that the steady state level of inhibitory RNA is much greater than the number of plasmids. The third and fourth assumptions enable the inhibitory RNA to be considered as being at a quasi-steady state level.

Given P copies of plasmid, the third and fourth assumptions above mean allow for the steady state level of inhibitory RNA R to be approximated by kP, where k is a large proportionality constant.

Then, assuming fast binding and unbinding of the RNA to and from the plasmid with some dissociation constant K_d, the following equations describing dissociation and mass conservation must hold:

$$\begin{aligned} K_d &= \frac{[P_f][R]}{[PR]}, \\ [P] &= [P_f] + [PR]. \end{aligned} \tag{4.1}$$

Here, $[P]$ denotes the concentration of P, so $[P] = \frac{P}{V}$, where V is the cell volume. The variable P_f denotes the number of free plasmids, while PR is the number of plasmid-RNA complexes, which have to add up to the total number of plasmids. Solving these two equations yields the following expression for $[P_f]$:

$$\left[P_f\right] = \frac{[P]}{K_d + [R]}. \tag{4.2}$$

Here, since $k \gg 1$, $[R]$ will be mostly unaffected by its binding to the plasmid, so substituting the steady state expression of R gives the following expression for $[P_f]$:

$$\left[P_f\right] = \frac{[P]}{K_d + k[P]}. \tag{4.3}$$

The initiation rate of plasmid replication is assumed to be proportional to the amount of free plasmids P_f, so multiplying both sides by the volume and re-arranging variables gives

$$P_f = \frac{\frac{1}{K_d}}{1 + \frac{[P]}{\left(\frac{K_d}{k}\right)}} P. \tag{4.4}$$

Since the propensity of plasmid replication is assumed to be proportional to the number of free plasmids P_f, the variables can be re-arranged to write down the following expression for the replication propensity, where the parameters have been combined into two parameters, β and K:

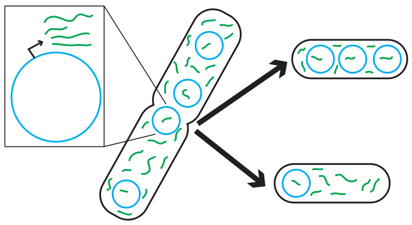


Figure 4.6: Plasmid-based transcription with partitioning. Each plasmid (blue) constitutively transcribes RNA molecules (green). Both plasmids and RNA's are partitioned between daughter cells during cell division.

$$\text{Replication Propensity} = \frac{\beta}{1+\frac{[P]}{K}}P. \tag{4.5}$$

A deterministic analysis of this plasmid replication rate can be performed. To do this analysis, assume that the cell volume is growing at a standard exponential rate with

$$\dot{V} = \alpha V. \tag{4.6}$$

Then, the dynamics of $[P]$ can be computed:

$$\begin{aligned}\frac{d[P]}{dt} &= \frac{d}{dt}\left(\frac{P}{V}\right) = \frac{V\dot{P}-P\dot{V}}{V^2} \\ &= \frac{1}{V^2}\left(PV\frac{\beta}{1+\frac{[P]}{K}} - \alpha PV\right) \\ &= \frac{P}{V}\left(\frac{\beta}{1+\frac{[P]}{K}} - \alpha\right).\end{aligned} \tag{4.7}$$

Setting the derivative equal to zero and solving gives the steady state value for the plasmid concentration,

$$[P]_{\text{eq}} = K\left(\frac{\beta}{\alpha}-1\right). \tag{4.8}$$

If volume is measured in units of cellular volume, then the average plasmid concentration can be thought as the steady state plasmid copy number. Note that $\beta > \alpha$ is required in order to have a non-negative steady state plasmid concentration. This is because the maximum rate of plasmid production must at least be able to keep up with the cell growth rate in order for the plasmid to be maintained.

4.5.2 Simulating Plasmid Replication and Gene Expression in Single Cells

Using the model of plasmid replication derived in the previous section, a model of plasmid replication combined with transcription can be used to compute the variability in mRNA levels between cells in a lineage simulation. In the model, there is one plasmid species, which replicates itself and also constitutively transcribes a mRNA. It is possible to look at the plasmid copy number and mRNA levels in a cell lineage over time as well as the plasmid copy number distribution across a population of cells at the end of the simulation. The full model used for producing the simulation is available in Appendix A.2. However, the model is tuned to produce a mean plasmid concentration of 10 nM, and the cell division time is the same 33 minutes as in the previous section.

The simulation is performed for 500 minutes and the plasmid distribution is empirically calculated using a final population size of 2048 cells. The run time for this simulation to compute a total of 4095 cell traces is less than two seconds on a standard desktop computer without using parallel processing.

As shown in Figure 4.7, the copy number at the end of the simulation has a wide distribution with a mean of about 15 copies per cell. This is expected because the mean concentration should be about 10 nM for the plasmid and the mean cell volume will be around 1.5 volume units. There is a slight peak in the distribution at a copy number of zero. This is because if a cell loses all its plasmids, it will continue dividing but its future descendants will never be able to recover the plasmid.

The distribution of plasmid and mRNA concentrations can also be plotted. In this case,

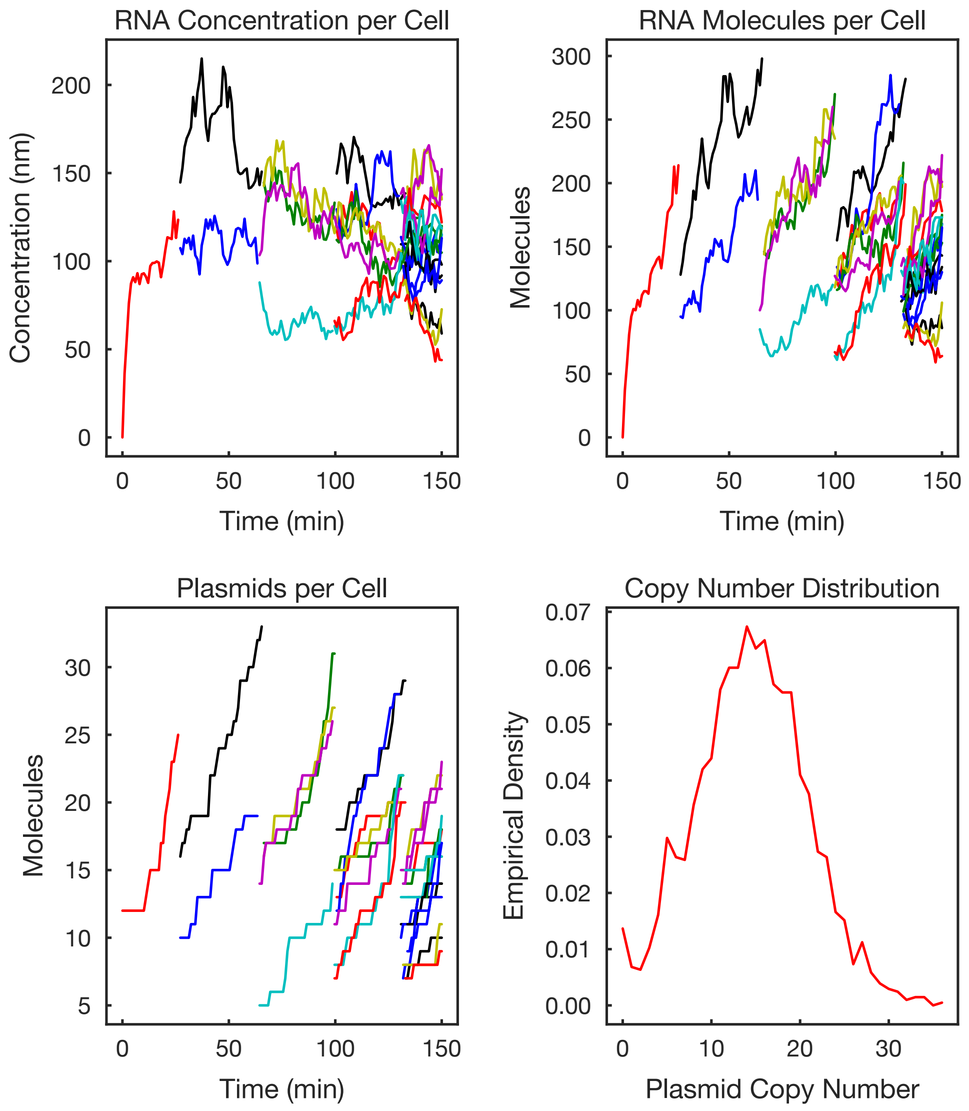


Figure 4.7: A simulation of plasmid replication and transcription over a cell lineage. The first three plots show trajectories of RNA and plasmid counts and concentrations over time. The last plot shows the distribution of plasmid copy number over 2048 cells at the end of the simulation.

the copy number is divided by the cell volume at the end of the simulation before plotting.

The expected plasmid concentration is 10 nM and the expected mRNA concentration is

93.45 nM. The results can be seen in Figure 4.8.

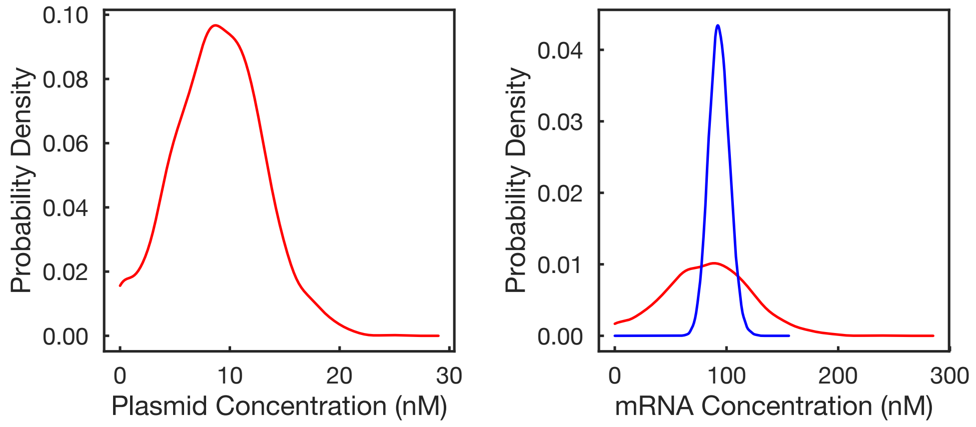


Figure 4.8: Distributions of plasmid and mRNA concentration across a cell lineage. The plasmid concentration is distributed around a mean of 10 nM. mRNA concentration is distributed around a mean of approximately 90 nM. The blue line shows the mRNA expression distribution if the plasmid concentration was exactly its mean value of 10 nM at all times.

The right panel of Figure 4.8 also shows a control where the plasmid concentration is assumed to be exactly controlled within the cell with no variability. In this case, the noise in mRNA expression is much smaller than in the case where the mRNA is expressed from the plasmid. The coefficient of variation in the plasmid based expression case is 0.55, while the coefficient of variation in the case with controlled copy number is 0.10. The XML code for the model where the plasmid copy number is exactly controlled is available in Appendix A.3.

4.6 Parameter Inference for Integrase Dynamics

In the previous sections, we demonstrated the capabilities of the bioscrape package for performing fast, flexible, and efficient simulations of biological circuits. In this section, we use the package's parameter inference capabilities to do parameter inference for a model of integrase dynamics based on *in vitro* experimental data. We first start by giving background on integrase systems and *in vitro* prototyping of biological circuits. We then describe the experimental procedure and the experimental data collected. Finally, we introduce the model and perform parameter inference on the model for both simulated data as well as the actual experimental data.

4.6.1 Background and Experimental Design

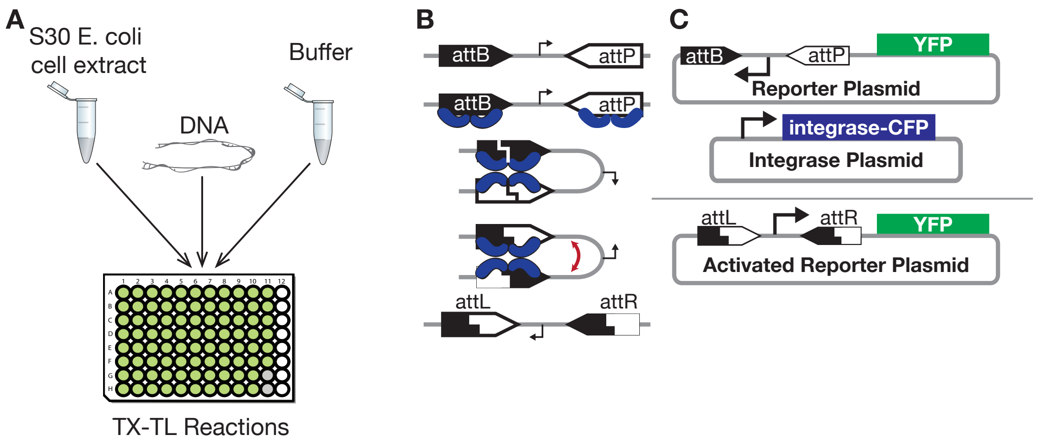


Figure 4.9: Testing serine integrase recombination dynamics using TX-TL. (A) The TX-TL system allows for prototyping synthetic circuits *in vitro* by adding DNA to cell extract and buffer. (B) Four serine integrases recombine attB and attP DNA sites to form attL and attR sites while reversing the DNA segment between the sites. (C) A constitutive integrase expression plasmid expresses integrase fused to cyan fluorescent protein (CFP), which flips a promoter on a a reporter plasmid and leads to yellow fluorescent protein (YFP) expression.

Both serine integrase systems and *in vitro* prototyping using cell free extracts are common tools in synthetic biology. Serine integrases are proteins that can recognize and recombine two specific target DNA sequences [29, 83]. Depending on the original directionality of the target sites, the recombination causes the segment of DNA between the target sites to either be excised or reversed. Figure 4.9B depicts the process by which four serine integrases bind to attB and attP DNA recognition sites and recombine them into attL and attR sites. In synthetic biology, this functionality has been leveraged to build synthetic gene circuits for state machines [77], temporal event detection [36], and rewritable memory [7]. However, existing applications of integrases rely on their digital behavior over long time scales, and not much is known about the dynamics of their action upon DNA.

One way to assay the dynamics of integrase DNA recombination is to test an integrase system using TX-TL, an *E. coli* cell extract *in vitro* system for testing and protoyping synthetic gene circuits [80]. Plasmid or linear DNA encoding the genes in a synthetic circuit can be added to a TX-TL master mix to prototype genetic circuits outside the cell as depicted in Figure 4.9A. In this case, we can create a simple synthetic circuit involving constitutive integrase production and reporter expression following DNA recombination to assay DNA recombination as a function of integrase levels. The circuit consists of two plasmids as shown in Figure 4.9C. On the first plasmid, the integrase plasmid, we constitutively express Bxb1, a commonly used serine integrase, as a part of a fusion protein in which Bxb1 is fused to CFP (cyan fluorescent protein). This allows us to use CFP fluorescence to measure the amount of Bxb1 present in the TX-TL reaction. The second plasmid is a reporter plasmid in which a promoter initially pointing away from a yellow fluorescent protein (YFP) gene can be reversed by integrase DNA recombination to point towards the YFP gene, which leads to production of YFP. Therefore, YFP expression can be used to infer when DNA recombination has occurred.

4.6.2 Experimental Results

Using automated acoustic liquid handling, we varied the level of integrase plasmid and reporter plasmid between 0 and 1 nM across 100 different TX-TL reactions. Each reaction contained integrase and reporter plasmid both independently at one of five concentrations of 0 nM, 0.25 nM, 0.50 nM, 0.75 nM, or 1 nM. Therefore, there were 25 possible combinations of concentrations of the two plasmids. Four replicates were done for each combination of concentrations, yielding a total of 100 TX-TL reactions. The reactions were incubated at 37 degrees Celsius, and CFP and YFP fluorescence were collected every 5 minutes for each reaction using a plate reader. Using a previously performed calibration of fluorescence to concentration, we were able to convert the fluorescence measurements for CFP and YFP to actual concentrations in nM for each fluorescent protein. Notably, the CFP concentration allowed us to measure the concentration of Bxb1 integrase in the reaction.

In Figure 4.10A, the full experimental data is presented. The dots represent actual data points, and the solid lines represent the median of 4 replicates. In Figure 4.10B, the median expressions are plotted in columns corresponding to fixed levels of reporter plasmid. As expected, the first row shows that integrase expression increases as integrase plasmid is increased. It is also clear from the second row of Figure 4.10B that reporter expression generally begins sooner and ends at a higher level when there is more integrase expression.

4.6.3 Model of Integrase Recombination

In order to estimate parameters for the integrase data presented in the previous section, we needed a model of integrase recombination of DNA. As a first cut, we created a simple model of integrase dynamics consisting of three reactions: integrase production, DNA recombination, and reporter production. As TX-TL is a bulk environment, we chose to use a deterministic model for our system, which we easily set up using the bioscrape package.

In Table 4.2, we describe the species in the model. These species are then used in the following set of ODE's that describe the integrase recombination dynamics in the model:

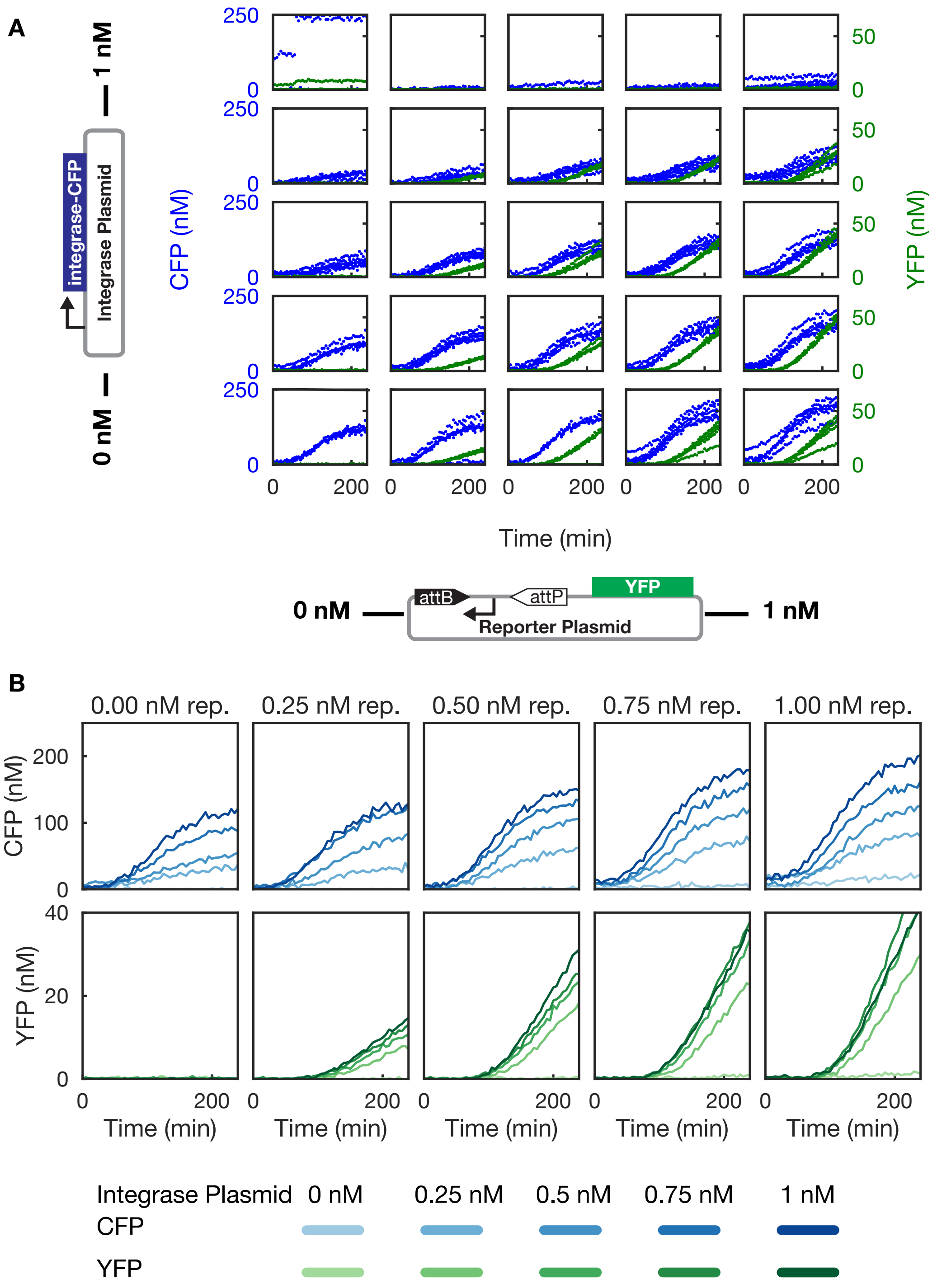


Figure 4.10: Experimental results for integrase testing. (A) Both integrase plasmid and reporter plasmid were varied from 0 to 1 nM and fluorescence data was collected for 4 hours. The dots are actual data points and the solid lines are the median of 4 replicates. (B) The median fluorescence trajectories plotted for fixed amounts of reporter plasmid. The reporter turns on sooner when more integrase is expressed.

Table 4.2: Species and parameters in the simple model of integrase recombination.

Variable	**Species**
I	Integrase-CFP (nM)
A	Activated reporter plasmid (nM)
R	Unactivated reporter plasmid (nM)
Y	YFP fluorescent reporter (nM)
I_{pl}	Integrase plasmid (nM)

Parameter	**Description**
k_I	Rate of integrase production (nM integrase per minute per nM integrase plasmid)
f	Maximum rate of integrase flipping of DNA (nM activated plasmid per nM reporter plasmid per minute)
K_f	Hill threshold for integrase activation (nM integrase)
n	Hill coefficient
k_Y	Rate of reporter production (nM reporter per minute per nM activated plasmid)

$$\begin{aligned}
\dot{I} &= k_I I_{\mathrm{pl}} \\
\dot{A} &= fR\frac{\left(\frac{I}{K_f}\right)^n}{1+\left(\frac{I}{K_f}\right)^n} \\
\dot{R} &= -\dot{A} \\
\dot{Y} &= k_Y A.
\end{aligned} \tag{4.9}$$

Equation 4.9 contains the ODE's for the simple model. Integrase is produced at a constitutive rate, where I_{pl} is the concentration of integrase plasmid and varies across experiments. The conversion of reporter plasmid to activated reporter plasmid is governed by a Hill function that allows us to model the cooperativity and activation threshold for the integrases in a simple way. We also assume that the DNA recombination reaction is first order in reporter plasmid. Finally, we assume that reporter is produced at a rate proportional to the amount of activated reporter plasmid. While varying the integrase plasmid changes the value of I_{pl} in the model, varying the reporter plasmid changes the initial condition for R.

Using representative values for the model, we created a simulated version of Figure 4.10B using the model. The plot is given in Figure 4.11, and there are some qualitative differences between integrase expression in the simulations and in the experimental data. Namely, while in the model the expression of integrase increases linearly with a slope proportional to the amount of integrase plasmid, in the experimental data, integrase expression only increases after a delay and then levels off after about two hours. This behavior is common in cell free extracts due to depletion of resources, and this effect should be included in a future more detailed model of the system. The full XML model for integrase dynamics and the numerical parameter values used in simulations are included in Appendix A.4.

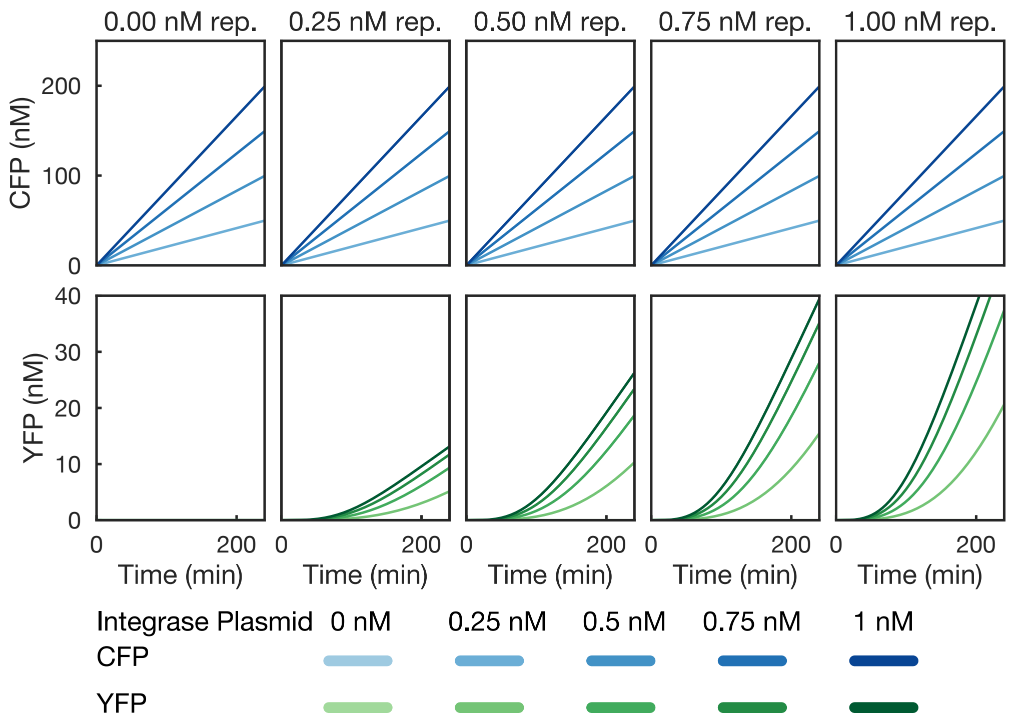


Figure 4.11: Simulations of integrase and reporter expression using the model qualitatively match experimental data (Figure 4.10B).

4.6.4 Parameter Inference for Integrase Dynamics

Using the model given in Equation 4.9, we attempted to perform parameter inference using bioscrape to fit the model parameters to both the simulated data from Figure 4.11 as well as the experimental median data from Figure 4.10B. Fitting the model to simulated data was a computational test of the identifiability of the model from the collected data. If a simulated version of the data were uninformative about parameter values in the models, then the real data would not be informative about the parameters either.

In bioscrape, parameter inference is performed in a Bayesian setting. In the Bayesian setting, prior beliefs or knowledge about the parameter is updated using the observed data to produce a posterior probability distribution over the parameter values. The Bayesian posterior distribution provides more information than other parameter inference techniques that produce a single estimate of the parameter values. For example, the shape of the posterior distribution can show how identifiable each parameter is, and whether parameter values are correlated with each other.

In the Bayesian setting, the prior knowledge about the parameter vector θ is encapsulated into a prior probability distribution $p(\theta)$. If nothing is known about the parameters, it is common for the prior distribution to be a uniform distribution over a large interval of parameter values, which is the type of prior distribution used here. Additionally, a likelihood function $p(y|\theta)$ gives the probability of observing the data y given a set of parameters θ. In this case, the forward model is an ODE model with Gaussian measurement noise, so the log of the likelihood function is proportional to the squared error between the data and model simulation. The goal of Bayesian inference is to find the posterior distribution $p(\theta|y)$, which gives the updated probability of each parameter set θ given that y was observed. However, Bayes' rule connects the posterior distribution to the prior distribution and likelihood function as follows:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}.$$

In this equation, $p(y)$, the probability of observing the data y, is a constant that is often difficult to compute, and so a more useful way to write the above equation is the following:

$$p(\theta|y) \propto p(y|\theta)p(\theta).$$

That is, the posterior distribution is proportional to a product of the prior distribution and likelihood function. While the posterior probability is hard to compute directly for a given θ, the prior probability and likelihood are straightforward to calculate for a given θ. This means that we can compute a quantity that is proportional to the posterior probability of a parameter set θ.

This fact can be used in conjunction with Markov chain Monte Carlo techniques to computationally produce samples from the posterior distribution $p(\theta|y)$. Once a satisfactory number of samples are drawn from the posterior, one can then compute statistics on the posterior distribution samples such as calculating the parameters' means and correlations.

The parameter inference code in bioscrape allows a user to enter a set of experiments into a likelihood function as well as specify a prior distribution on parameters. This information is then passed to an off the shelf ensemble Markov chain Monte Carlo package that generally works well on parameter inference problems [22, 28].

Figure 4.12 contains the posterior distributions for the parameters obtained after performing parameter estimation. From Figure 4.12A, it is clear that for simulated data, the true parameters are clearly identifiable from the simulated data. This suggests that the collected data should be informative about the parameter values.

Figure 4.12B contains the posterior parameter estimates from parameter estimation on the experimental data. In this case, most of the parameter distributions are again strongly peaked, except for the integrase throughput rate f. However, it is notable that the distribution for f is negligible for $f < 1$ and essentially uniform for $f > 1$. This suggests that the integrase throughput is much faster than gene expression. Because we are dependent on gene expression of the fluorescent reporter to infer when DNA recombination occurs, this posterior distribution suggests that we cannot exactly identify how fast the DNA recombination is, because the DNA recombination occurs much faster than the following gene expression.

The other parameters we are interested in are the Hill coefficient and activation threshold for integrase activity. These estimates are given in Table 4.3 along with their confidence intervals based on the posterior distributions. We found that Hill activation threshold was

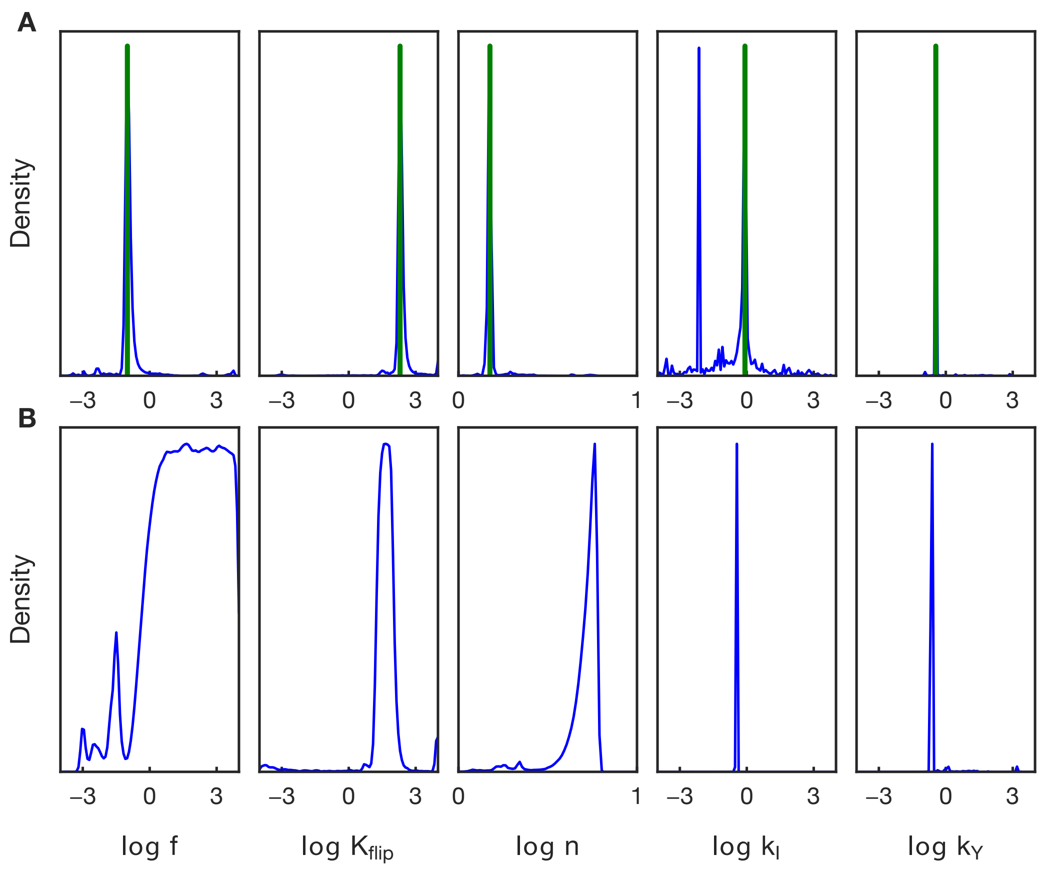


Figure 4.12: Posterior parameter estimates from MCMC. (A) Parameter distributions (blue) for the simulated data are strongly peaked around the true parameter values (green). (B) Parameter distributions for the experimental data from Figure 4.10B.

on the order of tens of nanomolar, which would correspond to an *in vivo* concentration of a few dozen molecules per cell. We found a Hill coefficient of 5.4, which was surprising, because four integrases combine to recombine DNA, so we expected the Hill coefficient to be no more than 4.

We also investigated the correlations between different parameters in the posterior. Because k_I and k_Y were tightly identified, we did not consider correlations involving those two parameters. Notably, we found positive correlations between $\log f$ and $\log K_{flip}$ and $\log f$ and $\log n$ respectively of 0.29 and 0.47. The correlation between $\log n$ and $\log K_{flip}$

Table 4.3: Estimates for integrase Hill coefficient and activation threshold.

Variable	**Median**	**16th to 84th percentile interval**
Hill coefficient n	5.4	(4.4,5.8)
Activation threshold K_f (nM)	45	(22,91)

was insignificant (0.02). The positive correlations with f are unsurprising, because if n or K_{flip} increases, it increases the effective delay time before integrase recombination can occur, which means that the recombination throughput f must be faster as well in order to generate the same dynamics of reporter expression.

In addition the simple Hill function based model of integrase recombination investigated here, we also considered a more mechanistically motivated model of recombination in which integrase molecules dimerize and bind to attB and attP DNA recognition sites in a reversible manner. Only plasmids with both the attB and attP site bound by integrase dimers are then able to undergo recombination. This model produces an effective delay in recombination by explicitly modeling the multiple binding steps required for integrase functionality. However, a parameter inference approach using this model showed that the model was not identifiable. The results are presented in Appendix B. The model XML is presented in Appendix A.5.

4.7 Discussion

The advent of increased computational resources and high throughput data collection for biological circuits has made quantitative modeling and parameter estimation for biological circuits more feasible. Since the most attractive parameter estimation techniques rely on Bayesian inference and Markov chain Monte Carlo (MCMC), it is important to have a simulator that can perform fast forward simulations of the model. Additionally, this simulator must be able to produce the same types of data that are observed in standard biological assays such as flow cytometry or fluorescence microscopy. Also, as models often need to be tweaked to fit the data, it should be easy to change the model or the way the model is simulated (e.g. switching from a deterministic to a stochastic simulation).

The bioscrape package addresses all of these issues. The flexible XML based language for model specification allows a user to easily make modifications to a biological circuit model by simply spending a minute editing a text file. The flexible Python based library for performing simulations allows for easily swapping between deterministic and stochastic simulations as well as consideration of other common effects in biological circuits such as cell growth and division and delays. Finally, because this package is written in Cython, its speed is comparable to the speed obtained using C code.

Performing simulations that incorporate effects like cell growth and division and delay can provide insight into the behavior of biological circuits. For example, in this chapter, it is demonstrated that in an exponentially dividing colony of cells, delays in gene expression can lead to a lower steady state protein level. This has implications for selection of fluorescent proteins for use in exponentially growing colonies. This chapter also demonstrates how transcribing a gene off a plasmid with copy number fluctuations will lead to more noise in expression than if the copy number of the plasmid were controlled exactly.

However, the ultimate aim of this package is to provide tools for doing parameter estimation for synthetic and systems biology. Here, we demonstrated the use of the bioscrape package to perform parameter estimation for both simulated and experimental data for integrase recombination dynamics in the TX-TL cell free *in vitro* system. As a result of this demonstration, we were able to estimate parameters for both the activation threshold and cooperativity of integrases that may be relevant for synthetic circuit design.

The fast simulators presented here will be the computational workhorse for more complex MCMC schemes for performing parameter inference for stochastic models of synthetic gene circuits. A future update to this report and the code will include inference methods and an experimental demonstration for a stochastic model.

Chapter 5

A Single Copy of a Single Gene is Sufficient for Rapid and Regular Oscillations

5.1 Introduction

Synthetic gene circuits do not come close to exhibiting the same robustness to noise as natural gene networks in cells. While cells can perform tasks like partitioning their genome between daughters perfectly across a wide range of conditions and across a population, the designed functions of synthetic gene circuits are often fragile to the effects of noise in gene expression.

In fact, most circuits are designed with deterministic considerations in mind such as oscillators [19], toggle switches [23] and logic gates [3]. However, single cells behave stochastically [20]. Stochastic fluctuations can decrease the performance of oscillators by destroying synchronization between cells in a population. Stochastic fluctuations can also spontaneously reverse the state of a genetic toggle, thereby decreasing the toggle's usefulness as a memory device. Designing circuits with stochastic performance in mind can allow for mitigating the effects of noise in gene expression. For example, in the case of oscillators, two oscillator designs might produce the same mean period and amplitude, but one design might be much less noisy than the other.

Oscillators in general have a rich history of application to synthetic biology questions. The first synthetic circuit engineering dynamics in cells was an oscillator [19], and oscillators have also been used as examples to demonstrate other ideas such as tunability [85], in vitro prototyping [69], temperature compensation [42], mitigation of loading effects [61], and effects of the intrinsic and extrinsic components of gene expression noise [93].

Specifically, both [69] and [73] took the original genetic oscillator design, the *repressilator*, reported in [19], and produced new improved versions that displayed increased robustness to noise. [73] leveraged tools from stochastic systems theory to infer that removing extrinsic effects such as active degradation of proteins and varying plasmid copy numbers would lead to a more robust oscillator. On the other hand, [69] used in vitro prototyping and control theory [33] to build a five gene version of the *repressilator* that also showed more robust and synchronized oscillations across a population. However, while stochastic performance was improved, in both cases the period was significantly lengthened as well.

Another significant genetic oscillator in synthetic biology is the activator-repressor oscillator originally published in [85], which uses two genes to generate oscillations that can be tuned by inducers. In [85], a single gene oscillator consisting of just a repressor with self negative feedback is also reported, and it is shown that the single gene oscillator's performance is significantly worse than that of the two gene oscillator, suggesting that the two genes are required for robustness. Furthermore, in [89], the period of an single gene delayed negative feedback oscillator is tuned by varying the length of the gene's intron. A longer intron creates a larger delay and thus a larger oscillatory period.

Here, we revisit and improve the single gene oscillator reported originally in [85]. Using a simple model similar to the one reported in [57], we predict that using a better repressor will not only improve the quality of oscillations but also change the period and amplitude.

However, the model also predicts that the original period of oscillations can be restored by changing the relative copy numbers of the genes in the circuit. This provides a roadmap to improving oscillations. First, we substitute an improved repressor; then we tune gene copy numbers to set the desired period.

5.2 Results

5.2.1 An Overview of the Single Gene Oscillator

The single gene oscillator, first reported in [85] and then modeled in [57], consists of two plasmids. The first plasmid contains a single repressor gene that represses its own production. A reporter gene that can be repressed by the same repressor is added into the cell on a separate plasmid. In the original single gene oscillator, the repressor used was lacI. Figure 5.1 shows the single gene oscillator with the repressor treRL [81] in place of lacI. Because there is some delay in production of treRL, both treRL and sfYFP (superfolder yellow fluorescent protein) [9] reporter can accumulate in the cell prior to repression occurring. This creates a burst of production of both the repressor and reporter genes. The endogenous protease ClpX then degrades both the reporter and repressor genes until the repressor level becomes low enough to begin another burst of expression. Because the repressor gene in the oscillator is actively degraded, it is possible to have periods of oscillations that are faster than the cell cycle time. However, active degradation has been shown to increase noise in oscillation period [73]. Furthermore, the repressor plasmid contains a p15a origin plasmid, known to have a much lower copy number than the ColE1 plasmid that contains the reporter gene [43]. Figure 5.1B shows a representative set of time lapse fluorescence microscopy images acquired for a micro-colony of oscillating cells, with the time between each frame being three minutes.

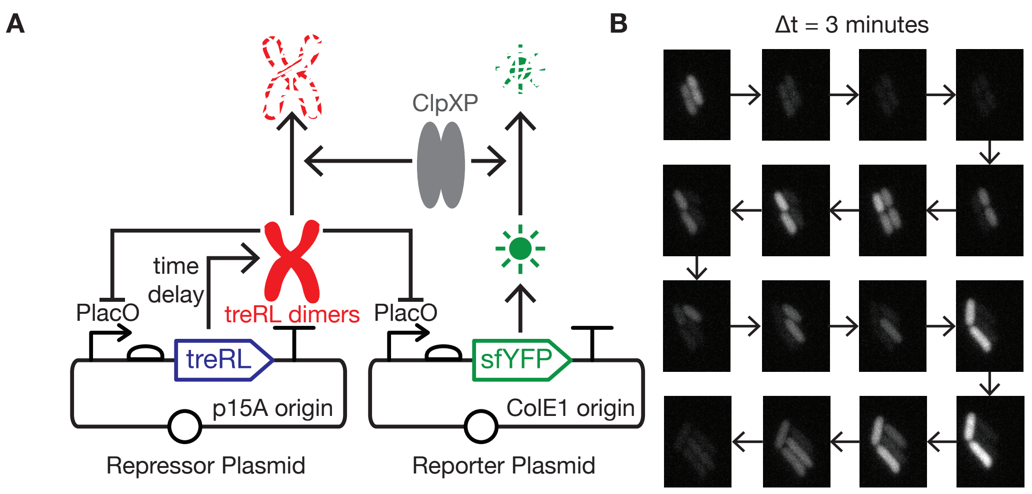


Figure 5.1: Delay in the production of a self repressing transcription factor can generate oscillations. (A) In a two plasmid system, the delay from treRL repressor transcription initiation to functional repression of its own promoter allows the treRL repressor and sfYFP reporter to build up. Active degradation of both repressor and reporter by ClpXP and cellular dilution reduces treRL and sfYFP levels until treRL production can fire again. (B) Fluorescence time lapse microscopy of sfYFP expression in a microcolony of cells. The frames, acquired every 3 minutes, show oscillatory fluorescence.

5.2.2 Modeling Guides the Design of an Improved Oscillator

In order to guide the design of an improved single gene oscillator, we created a simple model consisting of only production and degradation of reporter and repressor proteins along with cellular dilution. However, the model contains a delay between initiation of protein production and the final protein being completed. This model is similar to the model presented in [57] with the key difference being that we explicitly account for reporter production and degradation. This is because loading effects from the reporter gene on can change the period and amplitude of oscillations [14]. While loading effects typically negatively impact gene circuit performance, here we can leverage reporter loading to tune the period of oscillations.

The model assumes that the repressor in the oscillator exhibits zero leak and that the

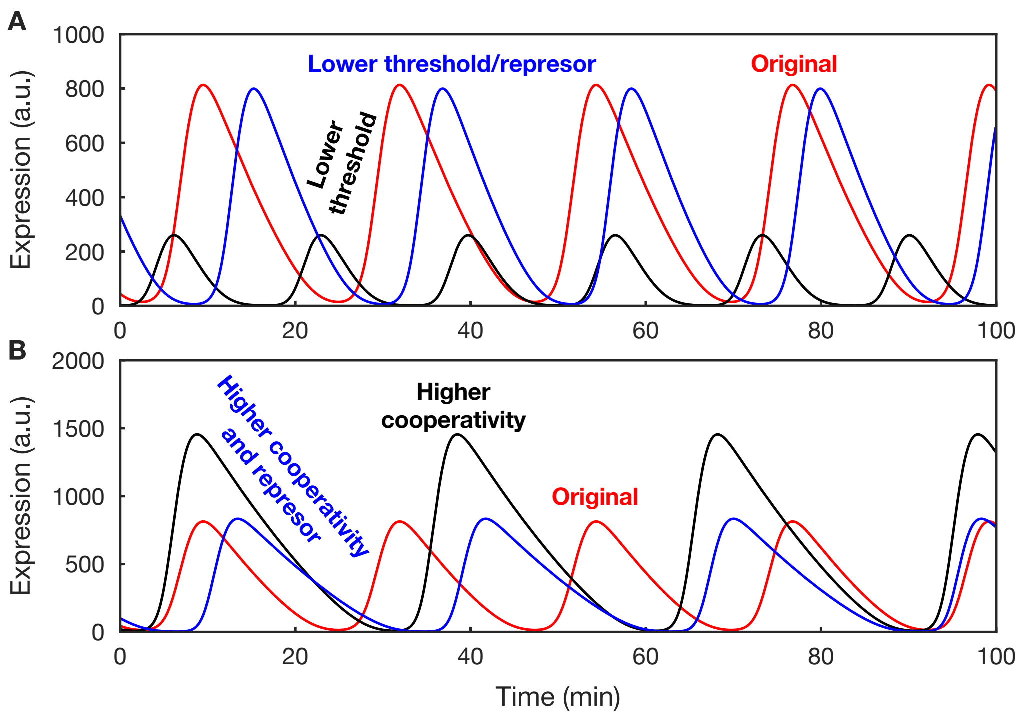


Figure 5.2: A simple model demonstrates that both decreasing Hill threshold and increasing cooperativity can improve oscillations. (A) Decreasing the Hill threshold reduces the trough and also makes the period and amplitude smaller. Decreased repressor expression recovers the original period and amplitude. (b) Increasing cooperativity reduces the trough and increases period and amplitude. Increased repressor expression recovers the original period and amplitude.

repressor behaves cooperatively with a Hill coefficient of $n = 2$. Additionally, the model assumes that reporter expression is much stronger than repressor expression, which is consistent with the copy numbers of the reporter and repressor plasmids as well as the ribosome binding sites used for the two genes.

Previous work in modeling of synthetic oscillators has shown that improving the cooperativity (Hill coefficient) or the binding strength (Hill threshold) of a repressor can improve oscillations [19, 33, 57, 73]. Intuitively, this occurs because both increased cooperativity and decreased threshold enhance the difference between the on and off states of the oscillation cycle. The cell is either expressing the repressor gene highly or not at all.

A common characteristic in non-robust oscillations is a high trough, or a peak-to-trough ratio that is close to unity. A low peak-to-trough ratio suggests that reporter gene expression is not fully switching off in the off state and suggests that the system might be close to the bifurcation point at which oscillations cease to exist. High troughs have been observed both in the original synthetic oscillator [19] as well in the lacI single gene oscillator (Figure 5.3A). In both of these oscillators, many cells in the population stop oscillating or never oscillate for long periods of time.

We believed that the key to improving the single gene oscillator was to lower the trough in the oscillations. The model of the oscillator predicts that both increasing cooperativity of the repressor or reducing the threshold for repression can reduce the trough in oscillations. However, these changes also affect the oscillation period.

In the regime of the model, where the repression threshold is strong and reporter expression dominates repressor expression, decreasing the repressor threshold reduces the trough of oscillations but also decreases period and amplitude. In order to recover the original period and amplitude, the repression must be weakened, and so tuning down the effective copy number of the repressor can recover the original longer period (Figure 5.2A). However, in this case, although the trough initially decreases due to the decrease in repression threshold, the trough increases again as repressor copy number is lowered. The hope is that this increase is not large enough to cancel out the initial decrease in trough.

On the other hand, increasing cooperativity of the repressor decreases the trough and at the same time increases the period and amplitude of oscillations. In order to restore the original period and amplitude, the repressor gene's copy number must be increased (Figure 5.2B), which decreases the trough even further. Thus, a repressor with increased cooperativity would be desirable over one with simply a lower repression threshold for improving oscillations.

5.2.3 Replacing lacI with treRL Improves Robustness of Oscillations

In order to fairly investigate the effect of changing repressor on the dynamics of the oscillator, it was important to keep the rest of the circuit constant. Fortunately, [60] developed a library of chimeric repressors that repress the lac promoter but are sensitive to different inducers. These chimeric repressors have been used in synthetic circuits, including in transcriptional AND gates [81]. In order to change only the repressor and keep the rest of the circuit constant, we could just replace lacI with a chimeric repressor that can still repress the same lac promoter. The results of [81] suggested that treRL, a chimeric repressor responsive to trehalose, might have a lower repression threshold than lacI. Thus, we decided to replace lacI with treRL in the circuit.

At this point, it is important to note that although the coding sequence for lacI was simply replaced with the coding sequence for treRL, translation and protein production is often sequence dependent even when using the same ribosome binding site [67]. However, because treRL is a chimeric repressor, the first 134 base pairs of the treRL coding sequence are identical to those of lacI, which means that any changes in translational efficiency should be minimal.

We also switched the reporter of the original lacI oscillator to a fast folding fluorescent reporter sfYFP (superfolder yellow fluorescent protein), which has been recently reported in [9]. Finally, we changed the lac promoter to the PlacO promoter used in [81]. The lac promoter and reporter gene remained constant across all experiments.

By replacing the lacI gene with treRL, we produced two oscillators, lacI p15a, and treRL p15a, where the repressor gene is on a plasmid with a p15a origin of replication. We performed fluorescence microscopy and image segmentation and tracking to characterize the temporal dynamics of fluorescence expression for both oscillators. The resulting trajectories were analyzed to find peaks and calculate distributions of periods and ampli-

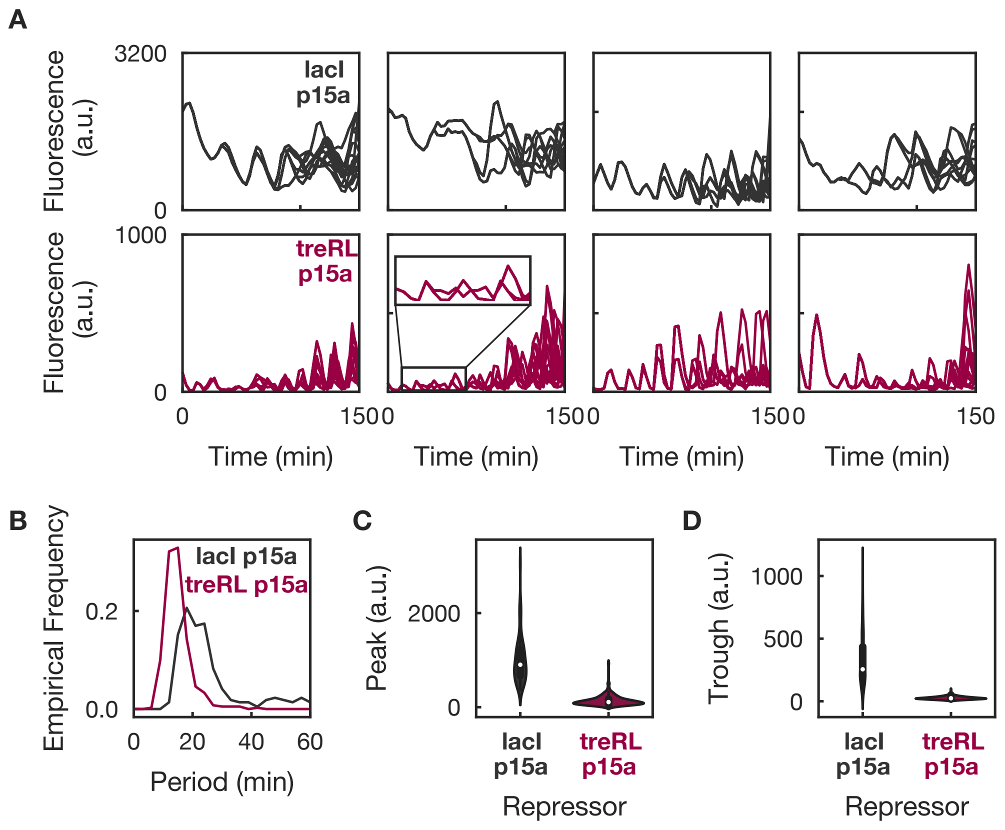


Figure 5.3: A treRL oscillator on p15a displays faster, more robust oscillations than lacI on p15a. (A) Example trajectories of both oscillators show that treRL trajectories are consistently oscillatory while lacI trajectories sometimes do not oscillate. treRL oscillations have a shorter period (B), lower trough (C), and lower peak (D) than lacI.

tudes.

We found that treRL oscillations occurred more consistently across a cell lineage. Figure 5.3A contains example fluorescence trajectories of single cell lineages for both the treRL p15a and lacI p15a oscillators. While some of the lacI trajectories do not oscillate and sit at a nonzero steady state, the treRL oscillations exhibit a much lower trough (Figure 5.3D) and all trajectories exhibit oscillations. In fact, many of the treRL p15a trajectories have an undetectable level of fluorescence over cellular autofluorescence in the trough.

Additionally, the period and amplitude for treRL p15a oscillations are much smaller than those for lacI p15a, with mean of 14.9 minutes for treRL p15a oscillations and 28.9 minutes for lacI oscillations. We believe that the treRL p15a oscillator, which is among the fastest synthetic oscillators, is notably robust given its speed.

Based on the modeling, this is consistent with treRL having the same cooperativity and a lower repression threshold than lacI. In accordance with the simple model, we predicted that reducing the copy number of the treRL gene would lengthen the period.

5.2.4 Reducing treRL Copy Number Slows Oscillations

In order to reduce the copy number of the treRL repressor, we replaced the p15a origin on the repressor plasmid with the lower copy psc101 origin, which is thought to have about half as many copies as p15a [43]. In doing so, we constructed two new oscillator variants, lacI psc101 and treRL psc101: lower copy versions of the original p15a oscillators. Furthermore, in order to reduce the copy number of the treRL gene even further, we integrated treRL into the genome using a common library for chromosomal integration [84]. We created genomic variants of the treRL oscillator with one or four copies integrated into the genome in the Phi186 integration site. These strains with genomically integrated treRL were then transformed with just the reporter plasmid to generate two further oscillator variants: treRL 4x and treRL 1x.

These variants were analyzed using microscopy and the results are summarized in Figure 5.4. Example trajectories of treRL psc101, treRL 4x, and treRL 1x oscillations show that all four variants generate robust oscillations across a lineage of cells. Furthermore, as the copy number decreases, the oscillations take a more asymmetric shape of an immediate burst followed by a long decay, which is consistent with the degrade and fire model proposed in [57]. The comparison between the lacI p15a and treRL p15a oscillators was consistent with a model of treRL having a lower Hill threshold that lacI. In this case, we expect to increasing period, increasing amplitude, and increasing trough with decreasing

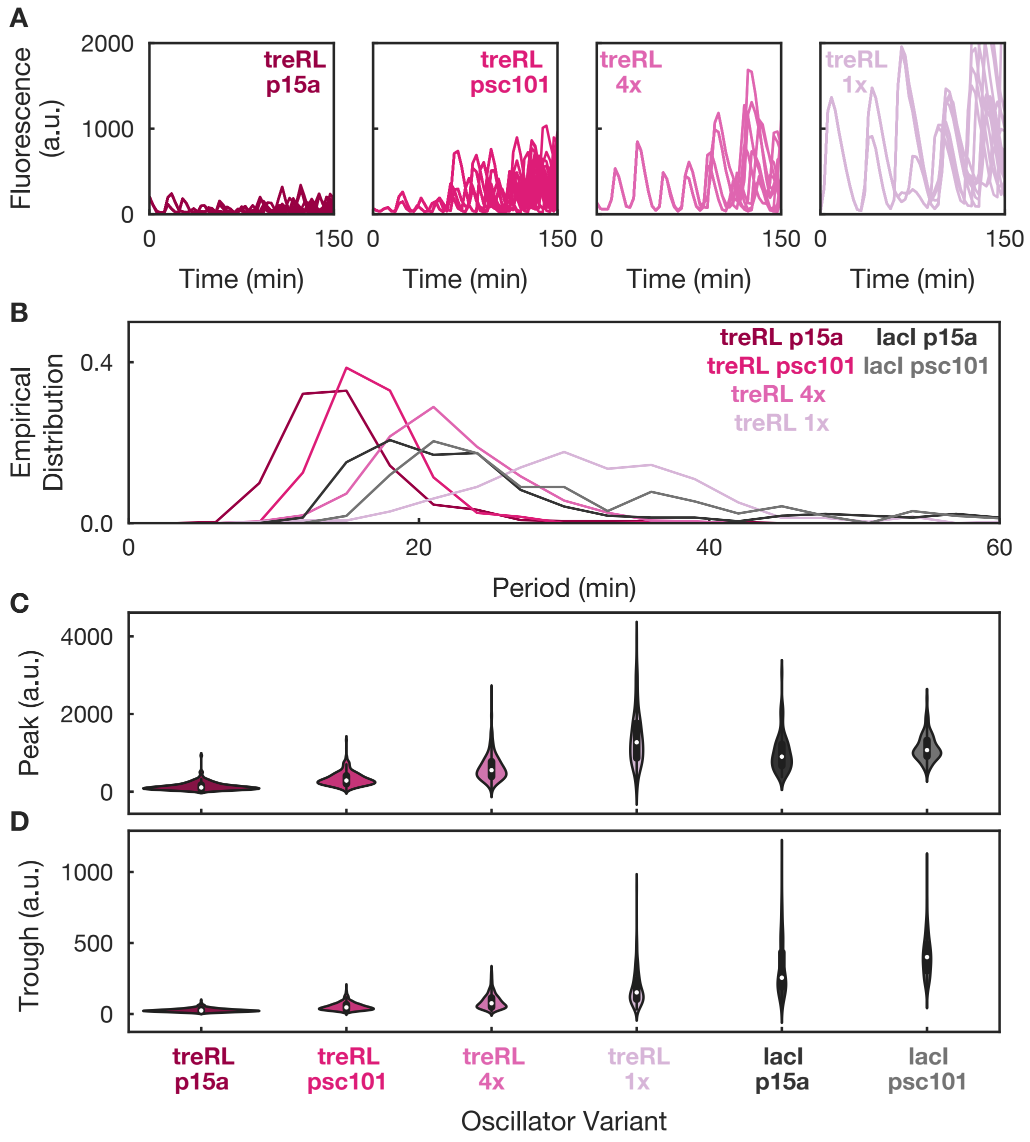


Figure 5.4: Reducing repressor copy number increases the period, amplitude, and trough of oscillations. (A) Example trajectories for treRL copy number variants show more robust oscillations at the lower copy numbers. (B) Periods for both treRL and lacI oscillators decrease with copy number. The trough and amplitude increase (C,D) as copy number decreases.

copy number, and that is exactly what is shown in Figure 5.4B-D. Additionally, in order to verify that the effects of copy number are not unique to treRL, we compare the lacI p15a and lacI psc101 oscillators and verify that the period, amplitude, and trough appear to increase for lacI as well.

5.2.5 Reducing Reporter Copy Number Speeds Up Oscillations

In addition to reducing treRL gene copy number, reducing the reporter copy number can also be shown by the model to reduce oscillation period (see Materials and Methods). In this case, the trough and amplitude should both get lower as well, because less reporter is being produced at all times. We tested this idea by constructing another oscillator variant, consisting of a single copy of treRL integrated in the genome like the treRL 1x variant, but we switched origin on the reporter plasmid from a ColE1 origin to a lower copy p15a origin. We call this oscillator the p15a reporter variant.

As expected, the reduction in reporter copy number generated a lower period, amplitude, and trough (Figure 5.5). We note that when inspected by eye, the p15a reporter oscillator generated the most consistent oscillations across the population, a fact that we will revisit computationally in Figure 5.7.

5.2.6 Simple Statistical Analysis Shows Oscillations Are Independent Period to Period

In order to better understand the noise in the oscillator dynamics, we then analyzed the relationships between successive periods and amplitudes across all trajectories. If intrinsic contributions to noise such as noisy gene expression of the repressor and reporter genes were the main driver of variation in period and amplitude, we would expect each successive period to be independent of the previous period and thus uncorrelated. If extrinsic factors such as number of polymerases, ribosomes, or proteases in the cell were a limiting factor, we would expect successive periods to be correlated, because cells with a set

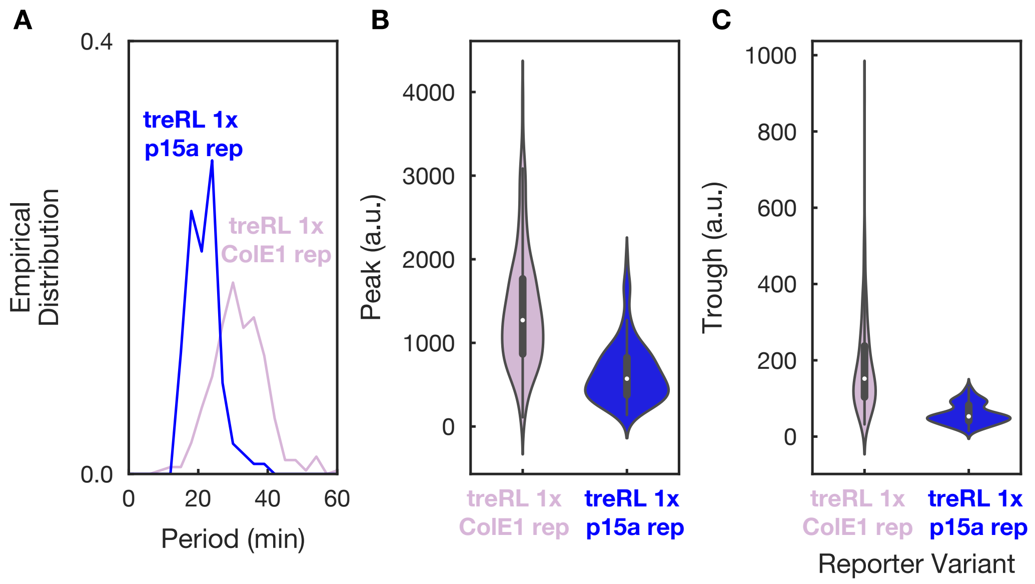


Figure 5.5: Reducing reporter copy number reduces period (A), amplitude (B), and trough (C) of oscillations.

of extrinsic factors favorable for generating long periods would be likely to generate long periods in the next cycle as well. The only case where the significance of extrinsic factors would not lead to correlated periods is if the extrinsic factors change on a similar time scale to the oscillation period, and so each successive period operates in an effectively independent set of extrinsic conditions.

When we investigated the relationship between successive periods and the relationship between an amplitude and its immediately preceding period, we found no meaningful correlations (Figure 5.6B,C). However, when we investigated the relationship between an amplitude and the immediately following period, we found that a higher peak tended to correlate to a longer period (Figure 5.6A). This is consistent with the degrade and fire model [57], in which the period is mostly determined by the amount of time required to enzymatically degrade a burst of repressor expression. The results in Figure 5.6 are shown for only the treRL 4x oscillator, but the results hold across all the oscillator variants. Fi-

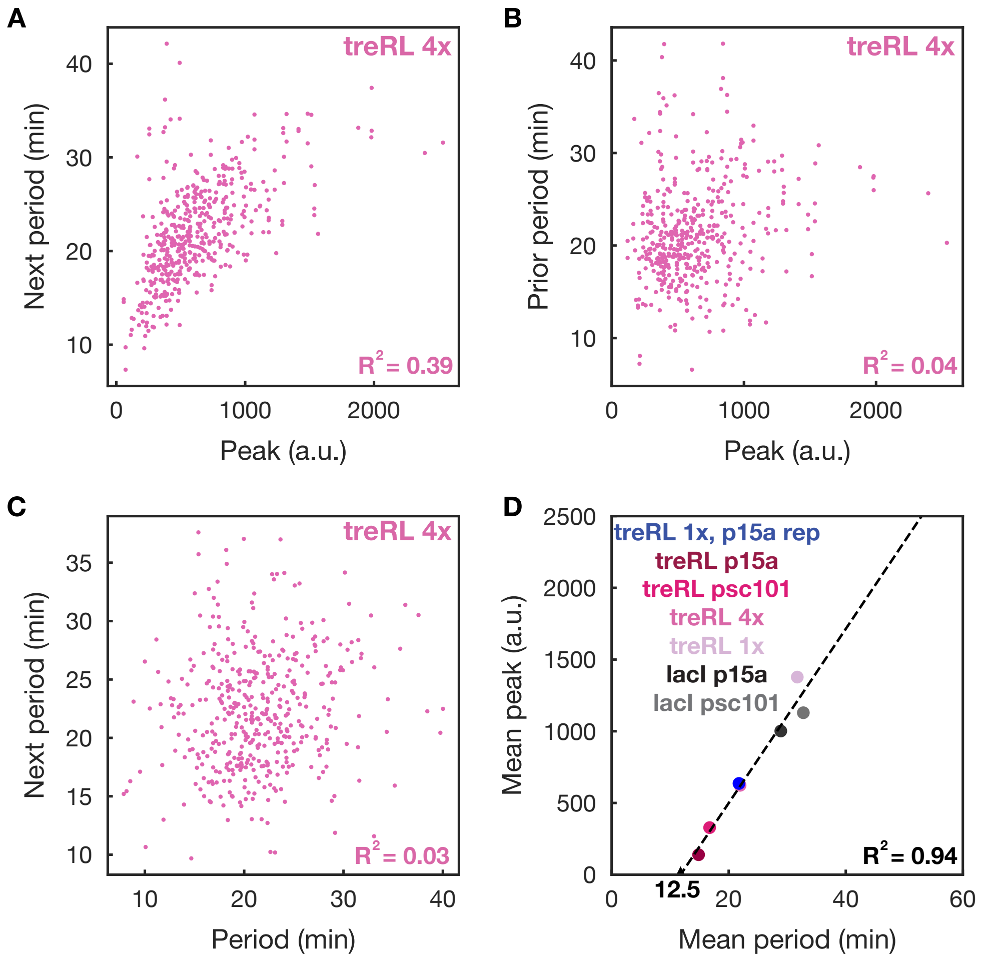


Figure 5.6: Analyzing successive periods and amplitudes reveals noise characteristics of the oscillator. (A) There is a medium strength relationship between a peak and its immediately following period. There is no relationship between a peak and its preceding period (B) or between two successive periods (C). (D) There is a strong linear relationship between mean period and mean amplitude across the variants, with a suggested minimum period of 12.5 minutes.

nally, the degrade and fire model also predicts that the period should be a minimum of twice the delay time in protein production with an additional factor that depends on amplitude. Figure 5.6D shows that the relationship between mean period and mean amplitude across variants is very strong, and by doing a linear fit, we can predict a minimum period of 12.5 minutes, suggesting that the delay time is on the order of 6.25 minutes. This value is consistent with previous work [85, 93].

5.2.7 treRL Oscillators Display Improved Synchronization Across Time and Cell Division

Finally, we sought to investigate whether our treRL oscillator variants displayed more synchronization across the population and over time than the origin lacI p15a oscillator. In order to ensure that the results were not simply an artifact of the analysis method, we measured synchronization in three different ways.

The first way is to measure the average difference in phase between daughter cells as a function of time since cell division across the population. We calculated the oscillation phase for each cell in the population, and for each cell division event, we kept track of the difference in phase between the two daughter cells at each successive time point until one of the daughter cells divided. We then computed a population average of the phase difference between daughter cells. While sister cells are initially perfectly in phase right after division, they may eventually fall out of phase with each other. Figure 5.7A shows the average phase decoherence in phase between daughter cells over time, with the treRL variants all performing similarly to each other and better than the lacI p15a variant.

The second way to measure synchronization is to compute the coefficient of variation of the period distribution. This value expressed as a percentage is termed the phase drift, and a phase drift of 14% was reported for the improved *repressilator* in [73]. Here, we calculate bootstrapped distributions of coefficient of variation of the period distributions for each of our best treRL oscillator variants compared to lacI p15a. We find that the treRL variants have much lower coefficients of variation, and notably the treRL psc101 oscillator has a phase drift of just 19% despite having a period of 16.7 minutes, which is 25 times faster than the oscillator reported in [73] (Figure 5.7B).

Finally, a third way to investigate synchronization is to compute the autocorrelation function over trajectories. In fact, while the previous two methods rely on calling peaks in trajectories, a procedure which can be inherently biased, calculating a sample autocorrelation is simply an arithmetic function of the data. However, in order to minimize the overwhelming influence of the earlier cells on a population autocorrelation, we de-weight contributions from earlier cells so that all cells' trajectories are evenly weighted in the computation. This led to much more reproducible autocorrelation functions between replicates (see Materials and Methods). Again, the autocorrelation functions demonstrate that the treRL oscillators are more synchronized than lacI, with each treRL variant displaying the characteristic sequence of decreasing peaks corresponding to a de-cohering oscillatory process (Figure 5.7C).

5.3 Discussion

To summarize, in this chapter we have improved on a previously reported single gene delay oscillator and created a set of new single gene delay oscillators that exhibit similar or faster periods, lower troughs, and stay more synchronized over time and across cell division. To improve the performance of the oscillator, we simply changed one part, the repressor protein, and then tuned DNA copy numbers to adjust the period and amplitude as desired.

We note that the synchronization properties of the treRL p15a oscillator are not as good as its lower copy number variants. There are at least two potential reasons why this could be the case. While the deterministic analysis suggests that turning up the gain with stronger repressor expression should only improve oscillations, a higher copy repressor also reduces the period and amplitude. At some point, the reduction in amplitude will

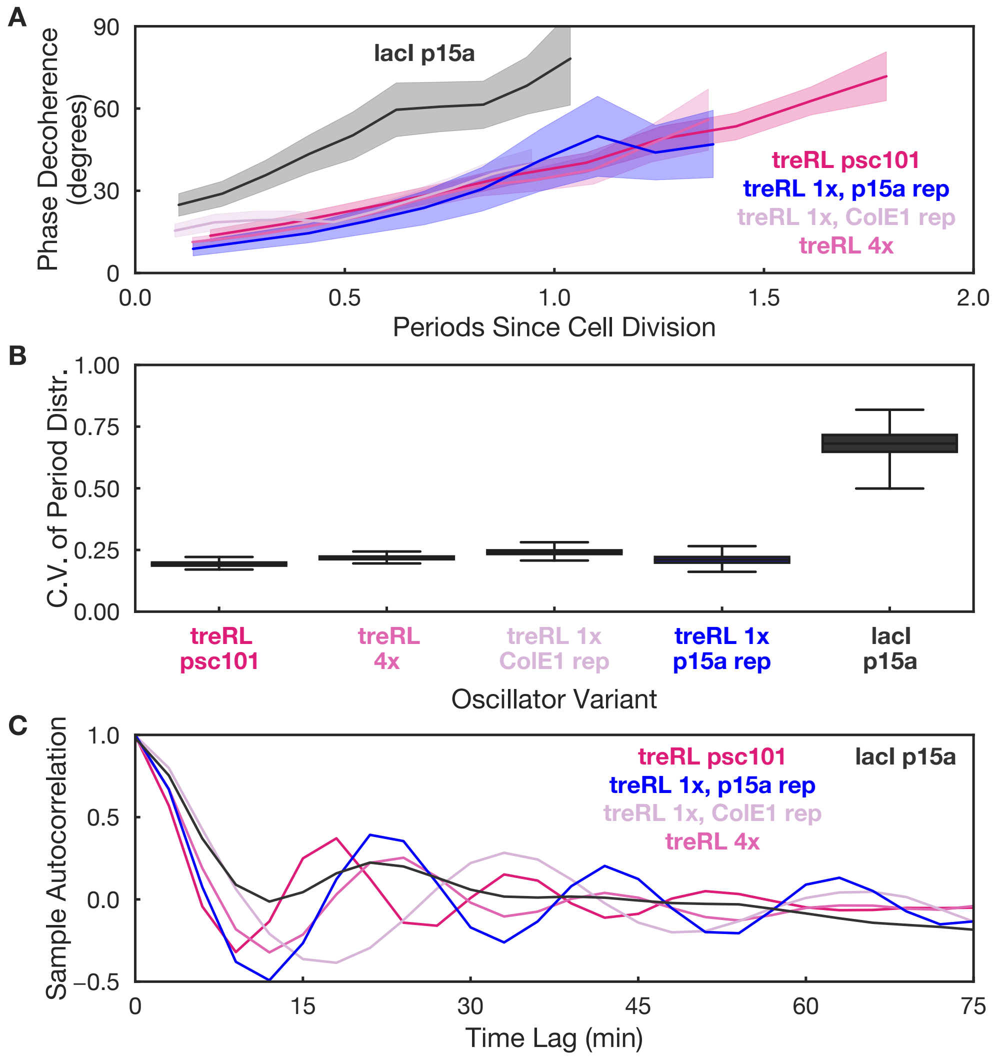


Figure 5.7: Analyzing the synchronization of oscillator variants. (A) Phase decoherence between sister cells as a function of number of oscillation periods since cell division. Error bars are the 16th and 84th percentile of 10,000 bootstrapped samples. (B) Coefficient of variation for the period distribution. Box plots show the error in the c.v. estimate across 1000 bootstrapped samples. Whiskers cover the entire bootstrapped sample. (C) Sample autocorrelation for each variant. The p15a reporter and treRL psc101 oscillators display the strongest autocorrelation functions, with the p15a reporter having a correlation of 0.4 after one period.

lead to low copy number stochastic effects that destroy the robustness of oscillations. Another possible explanation is the control of the gene copy number. While the genome is partitioned exactly at cell division, plasmids are known to have variation in their copy number both between cells and over time [72]. However, previous research has shown that the plasmids with a psc101 origin regulate their partitioning at cell division and maintain a more controlled copy number across a population [59]. Our results, which show that the phase drift is lower for oscillator variants that have treRL on the genome or on psc101, are consistent with an explanation of the noise in repressor copy number being responsible for the additional phase drift observed in the p15a oscillator.

Furthermore, we note that the phase drift for the treRL psc101, treRL 4x, treRL 1x, and p15a reporter oscillators is approximately same at 20%. We speculate whether this could be a fundamental limit on phase drift in the system, in which the inherent noise in ClpXP degradation prevents further improvement. Our best oscillator variant, the p15a reporter oscillator, has an autocorrelation of 0.4 at one period. This oscillator produces robust and synchronized oscillations with a single copy of the treRL gene, showing that even the simplest circuits can perform well if the parts are tuned optimally.

5.4 Materials and Methods

5.4.1 Modeling

The model for the oscillator is a simple ordinary differential equation model that incorporates delays in protein production by creating a series of intermediate states between initiation of protein production and the creation of final protein output.

The equations for the model are

Parameter	Value	Description
α	300 per minute	max production rate
P	12	repressor DNA copy number
M	60	reporter DNA copy number
K	0.03 (treRL) or 0.5 (lacI) molecules	repression threshold
γ	$\frac{\log(2)}{33}$ per minute	dilution rate
β	90 molecules per min	max degradation rate
R_0	100 proteins	degradation saturation constant
τ_x	5 minutes	delay time for repressor production
τ_y	5 minutes	delay time for reporter production
n	2	Hill coefficient

Table 5.1: Parameters for the ODE model describing the single gene oscillator.

$$\frac{dx}{dt} = \frac{\alpha P}{1+\left(\frac{x(t-\tau_x)}{K}\right)^n} - \gamma x - \beta \frac{x}{R_0 + x + y} \tag{5.1}$$

$$\frac{dy}{dt} = \frac{\alpha M}{1+\left(\frac{x(t-\tau_x)}{K}\right)^n} - \gamma y - \beta \frac{y}{R_0 + x + y}, \tag{5.2}$$

where x is the level of repressor and y is the level of reporter, with the parameter values given in Table 5.1.

In order to enforce the specified delay in production, we created a cascade of dummy states in-between the production initiation of treRL and lacI and the final arrival of a functional protein. The rate of passage through these rates and the number of states was tuned to approximate the desired delay. In the limit of an infinite number of intermediate states, this procedure approximates an exact fixed delay. We added states until it appeared that the dynamics no longer changed with additional intermediate states.

In the model, the repressor copy number, Hill threshold, and cooperativity can be tuned. The original curve in Figure 5.2A corresponds to the parameters in Table 5.1A for

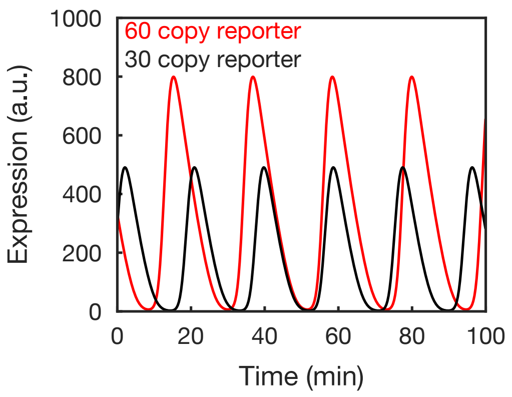


Figure 5.8: Reducing reporter copy number reduces the period and amplitude of oscillations.

lacI. Reducing the Hill threshold to the threshold for treRL generated the reduced threshold curve, and then further changing the repressor copy number from 12 to 1.7 produced the curve for reduced threshold and repressor.

In Figure 5.2B, the original curve again corresponds to the default parameters for lacI. Increasing the cooperativity to $n = 2.3$ generates the curve for higher cooperativity, and further increasing the repressor copy number to 48 creates the curve for higher cooperativity and repressor.

The reporter copy number can be tuned using the model as well. Using a repressor copy number of 1.7 and the Hill threshold for treRL, reducing the copy number of the reporter from 60 to 30 generates oscillations with a lower period and amplitude, as shown in Figure 5.8.

5.4.2 Plasmids and Strains

All experiments in this study were performed using the bacterial strain JS006, which was also used for the original single gene oscillator [85].

The reporter plasmid in the system contains a PlacO promoter [11, 81] driving expression of sfYFP [9] with the strong ribosome binding site BCD2 [67]. The sfYFP reporter is tagged with the native ssrA degradation tag (AANDENYALAA) for degradation by ClpXP. The reporter plasmid has a ColE1 origin and a Carb resistance gene. The reporter plasmid in all the variants is identical, except for the p15a reporter oscillator, which switches out the ColE1 origin for a p15a origin, as the name suggests.

The repressor plasmid in the system also contains a PlacO promoter driving treRL [81] or lacI expression using the Bujard RBS reported originally in [54]. The repressor plasmid contains a Kan resistance gene and a p15a origin in the p15a variants, and a psc101 origin in the psc101 variants. All the oscillator variants in this study have double antibiotic resistance to Kan and Carb.

In order to construct genome integrated treRL genes, we PCR'ed the treRL gene from the p15a plasmid and used golden gate assembly [21] to insert the treRL into the Phi186 site with Kan resistance in the genome using the method described in [84]. The resulting integrations were sequence verified from PCR products that were produced the P4 genome primer reported in [84] and a corresponding primer on the treRL gene.

In order to build the treRL 4x oscillator, a cassette plasmid containing one copy of the treRL gene was built and sequence verified. Using a standard starring assembly method in which BamHI and BglII sites can annihilate each other, we produced a cassette with two copies of the treRL gene. A second cycle yielded a cassette with 4 copies of the treRL gene, which was then inserted into the genome using the same method [84] and same site used for the single copy variant. Because sequencing of repetitive DNA is challenging, we were careful to sequence the treRL cassette gene, and then we performed all subsequent cloning steps using only digestion and ligation, which removes the possibility of mutations being introduced via PCR.

The strains with 1 and 4 copies of genome integrated treRL were transformed with the ColE1 reporter plasmid to create the treRL 1x and treRL 4x oscillators respectively.

Additionally, isothermal assembly using unique nucleotide sequences [91] was performed to swap out the ColE1 origin on the reporter plasmid for a p15a origin, the same p15a origin as the original repressor plasmid. This p15a origin reporter was transformed into the strain with a single copy of treRL integrated in the genome to create the p15a reporter oscillator variant.

5.4.3 Data Acquisition

In order to collect data, strains were streaked out onto a plate. Colonies were picked and grown overnight at 37 degrees Celsius in 5 mL cultures in M9CA media (Teknova M8010), which contains 1.0% glucose and casamino acids in addition to M9 salts. This media proved superior to LB because of its lower background in fluorescence imaging and its more repeatable results. Strains in the morning were diluted by a factor of 1000 to 5000 fold into fresh 5mL M9CA media cultures and outgrowth was performed for 4-7 hours until the cells reached an OD600 of at least 0.1 and no more than 0.4. Both the overnight culture and outgrowth step were performed in M9CA media with Carb and Kan antibiotics added. While Carb was used at its usual level of 100 μg/mL, Kan was used at half its usual concentration. We used 25 μg/mL of Kan, because we noticed that the variants with only a single copy of the Kan resistance gene integrated into the genome grew much slower at the normal kanamycin concentration than the oscillator variants with the resistance gene on a plasmid.

After outgrowth, the cells were diluted to an OD600 of 0.01 and seeded onto agarose pads containing M9CA media with only Carb and no inducers. No kanamycin was used in the pads in order to better equalize cell growth rate from experiment to experiment. We used the protocols for creating agarose pads, seeding culture onto pads, and setting up movies on pads described in [95]. We let our agarose pads dry at room temperature after seeding cultures, and then placed the pads onto 40 mm No. 1.5 Willco dishes (GWSt-5040) for microscopy. The microscope objective and sample were automatically temperature controlled to 37 degrees using a heater, and so we had to wait 15-20 minutes after placing the sample on the microscope for the temperature to reach and stabilize at 37 degrees.

To perform automated fluorescence time lapse microscopy, we used an Olympus IX-81 microscope with a ZDC autofocus and motorized stage that allowed reliable image acquisition at multiple positions over time. Notably, we used a Hamamatsu ORCA-Flash 4.0LT camera and Lumencor SOLA SE Light Engine. The light settings for fluorescence acquisition were 25% intensity for 400 ms of exposure, and one frame was collected every 3 minutes. All details of the microscopy were identical across day to day replicates and oscillator variants as well as control experiments.

To automate the collection of images, we used Micro-Manager software [17].

5.4.4 Image Segmentation and Tracking

In order to perform segmentation and tracking, we binned our 2048 x 2048 TIFF images to 1024 x 1024. We then fed them into the Super Segger software package [86], which automatically performed segmentation and tracking and background calculation for the cells over time. We inspected the resulting segmentation and tracking using Super Segger's own tool, and we found that the tracking in all cases was essentially perfect in every movie up to 2.5 hours. After 2.5 hours, some movies exhibited cells growing on top of each other, which compromised tracking quality, so we used 2.5 hours of segmented data (51 frames) in our data analysis. To correct for background, we used Super Segger's built-in background calculation. This removed the background due to the agarose pad and noise in the camera itself. However, cellular autofluorescence is another source of background. To correct for this background, we segmented and tracked a movie of just JS006 cells growing on an M9CA agarose pad with no antibiotics. The background adjusted median fluorescence of the autofluorescent cells, which we found to be 18.02 units, was subtracted from the median fluorescence observed in the oscillator variants.

To compute fluorescence trajectories for the oscillator variants, we took the median expression in the cell at each time and subtracted the 18.02 units for autofluorescence. In the treRL oscillators, the off state was so strong that sometimes the correction for autofluorescence would result in a negative fluorescence value. These negative values were increased to zero if encountered, so all fluorescence values in the fully background corrected trajectories are non-negative.

These background corrected fluorescence intensities were then compiled into a Lineage data structure compatible with our lab's software for parameter inference and simulation of biological models [87]. This data structure keeps track of each cell's time points, median fluorescence intensities, volumes, and parents and daughter cells.

5.4.5 Oscillation Analysis

We wrote our own Python library for analysis of oscillations across lineages.

5.4.5.1 Estimating Periods, Amplitudes, and Phase

In order to estimate periods and amplitudes for oscillations, we first identified peaks in trajectories using a method that finds peaks and troughs in the data set that must be separated by some minimum user-specified threshold [6]. In order not to bias the peak-finding algorithm towards performing differently on oscillator variants with higher or lower peaks, we instead took a logarithm of the fluorescence intensities, and we used the threshold parameter to enforce that any identified peak must be at least 1.5 times the expression of its adjacent troughs. The cutoff of 1.5 was identified using manual inspection of peaks overlaid onto trajectories.

A raw fluorescence cutoff of 25 was applied to all peaks, so identified peaks with a expression of 25 units or less were removed from the calculation. This removed spurious peaks at the low end of expression, an effect only seen in the treRL p15a oscillator.

The inter-peak distances were used to compute periods in multiples of three minutes, and the amplitudes of the peaks were saved as the amplitudes of oscillations. Additionally, the minimum value between two peaks was saved as the trough in the oscillation. Note that because of the branching structure of the lineage, a peak can be associated with up to three periods. A peak and the prior peak in the same trajectory can define one period, while in the future a peak may correspond to two separate peaks in two daughter cells.

In order to compute phase, we unfolded all the trajectories in the lineage to a single vector of time points for the entire movie, fluorescence values along the time points, and the ID of each cell in the trajectory. Thus each successive cell that appears in the trajectory must be a daughter of the previous cell. Once this was complete, we identified peaks for each trajectory and calculated phase for the trajectory as starting at 0 at the first peak, and increasing by 2π at each successive peak with linear interpolation. The phase before the first peak and after the last peak was left undefined (as NaN). Once the phase for each trajectory was computed, we calculated the phase for each cell by averaging over trajectories. That is, if a cell appeared in multiple trajectories, the phase of the cell would be the average of its phase in those trajectories. If undefined values were present in this scenario, they were excluded from the averaging operation.

This analysis pipeline generates the figures in the text that plot the distributions of periods, amplitudes, troughs, and phase difference between daughter cells.

5.4.5.2 Calculating the Sample Autocorrelation

We wanted to create a method for measuring synchronization that would function independently of biases in calling peaks. Thus, we computed a sample autocorrelation for each oscillator variant. The sample autocorrelation is a standard quantity that is used to understand temporal relationships in a signal. However, the interesting case here is that the signal in this setting is a branching signal. This creates a unique problem in which it is not clear how to calculate the sample autocorrelation. For example, it is not clear whether all instances of a given time lag should be pooled across the population, which allows for an estimation of a single correlation, or if trajectories should be unfolded, autocorrelations should be computed for each trajectory, and then the autocorrelations averaged to generate an autocorrelation function.

We found empirically that the second approach (unfolding trajectories and averaging computed autocorrelations) produced much more repeatable autocorrelations from run to run.

The second problem in the autocorrelation is that the starting few cells in the population have an outsized impact on the autocorrelation, because they appear in exponentially more trajectories than future cells. Thus, the measurements at a few time points in the initial cell can affect the entire sample autocorrelation across the lineage and create run to run variability. In order to reduce this variability and put all cells on an even footing, we used a weighting scheme when computing the autocorrelation.

To do this, we first noticed that if the experiment were to run for N generations, the mother cell would appear in 2^N trajectories, while final cells would appear in only one trajectory. Therefore, for each trajectory we weighted the sample means of the covariance and variance by powers of two corresponding to the generation of each cell. For example, pairs of time points involving the mother cell were weighted with a weight of 1, while pairs of time points in which the earlier cell was in the third generation would be weighted with a weight of 4. In this way, the final cells all contribute to their trajectory's autocorrelation exponentially more than the initial cells, but the initial cells contribute to exponentially more trajectories. This balances out the overemphasis on the initial cell's expression and leads to much cleaner and more reproducible autocorrelations. This method for calculating the sample autocorrelation is similar to the method used in [16]. However, the method used in [16] produces different results based on the ordering of trajectories, whereas the method described here does not depend on any trajectory ordering and always produces the same result.

Stated mathematically, suppose that all trajectories contain equally spaced time points at even intervals. Then suppose there are N time points, with y_i representing the mean corrected fluorescence at each point. That is, we assume that the mean of the fluorescence along the trajectories has been removed. Then, assume that d_i represents the number of cell divisions that have occurred since the beginning of the trajectory. The sample autocorrelation $c(\tau)$ for a time lag of τ time intervals is given as follows.

$$c(\tau) = \frac{\dfrac{\sum_{i=1}^{N-\tau} 2^{d_i} y_i y_{i+\tau}}{N-\tau \sum_{i=1}^{N-\tau} 2^{d_i}}}{\dfrac{\sum_{i=1}^{N} 2^{d_i} y_i^2}{N \sum_{i=1}^{N} 2^{d_i}}} \tag{5.3}$$

Chapter 6

Conclusion and Future Directions

While modeling of synthetic gene circuits has not yet found widespread application, it is our hope that the field is moving towards a future in which quantitative characterization and parametrization play a role in the design and implementation of synthetic gene circuits. The results from this thesis constitute a step towards that goal.

In Chapter 3 of this thesis, we developed conditions for identifiability of linear systems using a combination of time series and distributional measurements. Time series measurements for synthetic gene circuits are often limited to a small number of outputs, while distributional measurements can measure a larger number of outputs across a population of cells. For the theoretically tractable example of linear systems, we investigated whether the combination of dynamic information from time series measurements and the high-dimensional information from distributional measurements could improve system identifiability. We showed this to be the case both with theoretical results as well as with computational examples.

However, there are some remaining theoretical questions in this area. First of all, we developed necessary and sufficient conditions on the number of time series output measurements required to uniquely identify linear system dynamics. However, there was a large gap between the necessary condition and sufficient condition in terms of the number of measurements required, and a more careful analysis may yield tighter conditions with a smaller gap. Secondly, all of our analysis assumed that the system dynamics could be observed with perfect measurements for an infinite amount of time, so that the system covariance and output correlation functions could be observed exactly. Furthermore, we also only proved results for linear systems driven by white noise. A more detailed analysis accounting for finite sample sizes, measurement noise, and nonlinearity in the dynamics would yield results more directly applicable to actual modeling of synthetic gene circuits. However, these questions are significantly more difficult to tackle theoretically, and so we believe using computational methods to assess identifiability as in Chapter 4 is a more practical approach.

In Chapter 4, we described bioscrape, an open-source Python package that we have developed for fast and flexible simulation and parameter estimation for models of synthetic gene circuits. We showed that bioscrape can can simulate gene circuit dynamics deterministically or stochastically, and that bioscrape can simulate dynamics at the bulk, single cell, or cell lineage level. We then used bioscrape to perform parameter estimation for two different models of integrase recombination using experimental data collected in an *E. coli* cell extract. Because of bioscrape's flexible modeling framework, performing parameter estimation for two models of integrase recombination was as simple as describing each model using a human-readable XML modeling language, and then changing a few lines of parameter estimation code to switch models. We assessed identifiability for each model using simulated data sets, and we then ran the same code using the real data to perform parameter estimation. We found that a simple Hill function model was more identifiable and generated tighter posterior distributions than a more complicated full mechanistic model of integrase recombination.

Future work on bioscrape includes expanding the parameter estimation code to different types of data sets, such as cell lineage data collected from time-lapse fluorescence microscopy, or population distributions collected from flow cytometry. Ideally, a researcher should be able to input any type of data into bioscrape, specify a model and prior parameter distributions, and receive posterior parameter estimates. Another remaining task is to test and document the software more thoroughly. The end goal is to create a user-friendly piece of software that can handle heterogeneous data types and makes modeling simple, so that more researchers consider doing quantitative modeling of their data.

Lastly, on the scientific side, because bioscrape can simulate cell lineages, it also lends itself to computational analysis of phenomena that occur at the population level. For example, bioscrape could be used to investigate the phase decoherence observed in the single gene oscillator from Chapter 5. By simulating a model of a single gene oscillator at the cell lineage level, we could identify how different parameters affect phase deocoherence in a population of cells.

Finally, in Chapter 5, we described the model-guided design and construction of an improved set of single gene oscillators that demonstrate more regular and synchronized oscillations across a population of cells than a previously described single gene oscillator [85]. The results are a demonstration of model-guided design; predictable tuning of certain parts of the circuit improved oscillation quality and allowed for tuning of period and amplitude. The best oscillator variants we constructed also showed a phase drift of 19%, comparable to the most synchronized oscillator reported [73], showing that a single copy of a single gene is enough to generate regular oscillations.

Because the period of these oscillations is fast, and because the design is simple and requires only one gene, the natural next step would be to attempt to construct two orthogonal single gene oscillators in the same cell using two orthogonal repressors. A putative system of two fast and orthogonal oscillators in the same cell would enable more complex temporal regulation of cellular function. For example, some downstream functionality could activate when one oscillator pulses, but other downstream functionality could require both oscillators to pulse simultaneously, which is a rarer occurrence. In this way, the relative scheduling of different cellular processes could be tuned.

In order for synthetic biology to truly become an engineering discipline, it must be possible for design and modeling of synthetic gene circuits to reliably predict experimental observations. Such progress requires tighter integration between application, computation, and theory, so that theory and computation are developed with applications in mind, and applications can test out and refine theoretical and computational techniques. Here, we developed theoretical identifiability conditions (Chapter 3) that ideally will help circuit designers decide which outputs to measure for their synthetic gene circuit. We also developed software (Chapter 4) that we hope will make modeling and parameter estimation more accessible for synthetic gene circuits. Finally, we demonstrated that model-guided design can improve and tune a single gene oscillator in a predictable way (Chapter 5). As computation, theory, and application become more tightly connected, it is our hope that synthetic gene circuit design will progress towards a future in which a single iteration of the design-build-test cycle will yield a properly functioning gene circuit.

Appendix A

XML Models for Simulations

This section contains the XML models used for simulation and parameter inference in Chapter 4.

A.1 Full XML Model for Simple Gene Expression

```
<model>
<reaction text=”—” after=”—mRNA”>
    <propensity type=”massaction” k=”beta” species=”” />
        <delay type=”fixed” delay=”tx_delay” />
</reaction>

<reaction text=”mRNA—” after=”—”>
    <propensity type=”massaction” k=”delta_m” species=”mRNA” />
        <delay type=”none” />
</reaction>

<reaction text=”—” after=”—protein”>
    <propensity type=”massaction” k=”k_tl” species=”mRNA” />
    <delay type=”gamma” k=”tl_k” theta=”tl_theta” />
```

```
</reaction>

<reaction text="protein —">
    <propensity type="massaction" k="delta_p" species="protein" />
    <delay type="none" />
</reaction>



<parameter name="beta" value="2.0" />
<parameter name="delta_m" value="0.2" />
<parameter name="k_tl" value="5.0" />
<parameter name="delta_p" value="0.05" />
<parameter name="tx_delay" value="10" />
<parameter name="tl_k" value="2" />
<parameter name="tl_theta" value="5" />

<species name="mRNA" value="0" />
<species name="protein" value="0" />
</model>
```

A.2 Full XML Model for Plasmid Replication and Transcription

```
<model>

<reaction text="—plasmid" after="—">
        <propensity type="proportionalhillnegative" k="beta_plasmid" n="n"
                        K="K_plasmid" s1="plasmid" d="plasmid" />
        <delay type="none" />
</reaction>
```

```
<reaction text=”—mRNA” after=”—”>
    <propensity type=”massaction” k=”k” species=”plasmid” />
        <delay type=”none” />
</reaction>

<reaction text=”mRNA—” after=”—”>
    <propensity type=”massaction” k=”delta” species=”mRNA” />
        <delay type=”none” />
</reaction>



<parameter name=”beta_plasmid” value=”0.04200892003” />
<parameter name=”n” value=”1.0” />
<parameter name=”K_plasmid” value=”10”/>

<parameter name=”k” value=”3.0” />
<parameter name=”delta” value=”0.3” />

<species name=”mRNA” value=”0” />
<species name=”plasmid” value=”12” />

</model>
```

A.3 Full XML Model for Transcription with Exactly Controlled Copy Number

```
<model>

<reaction text=”—mRNA” after=”—”>
    <propensity type=”massaction” k=”k” species=”” />
```

```
        <delay type=”none” />
</reaction>

<reaction text=”mRNA—” after=”—”>
    <propensity type=”massaction” k=”delta” species=”mRNA” />
        <delay type=”none” />
</reaction>

<parameter name=”k” value=”30.0” />
<parameter name=”delta” value=”0.3” />

<species name=”mRNA” value=”0” />

</model>
```

A.4 XML Model for Hill Function-Based Integrase Dynamics

```
<model>

<reaction text=”—I” after=”—”>
    <delay type=”none” />
    <propensity type=”unimolecular” k=”k_I” s1=”I_pI” />
</reaction>

<reaction text=”R—A” after=”—”>
    <delay type=”none” />
    <propensity type=”proportionalhillpositive” k=”f” K=”K_f” n=”n” s1=”I” d=”R” />
</reaction>

<reaction text=”—Y” after=”—”>
    <delay type=”none” />
    <propensity type=”unimolecular” k=”k_Y” s1=”A” />
</reaction>

<species name=”A” value=”0” />
<species name=”R” value=”0” />
<species name=”I_pI” value=”0” />
<species name=”I” value=”0” />
<species name=”Y” value=”0” />

<parameter name=”f” value=”0.1”/>
<parameter name=”K_f” value=”200” />
<parameter name=”n” value=”1.5” />
<parameter name=”k_I” value=”0.83” />
<parameter name=”k_Y” value=”0.357” />

</model>
```

A.5 XML Model for Full Mechanistic Integrase Dynamics

```
<model>

<reaction text=”—If” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_I” species=”I_pI” />
</reaction>

<reaction text=”If+If—D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”I*I” />
</reaction>
```

```
<reaction text=”D—If+If” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdim” species=”D” />
</reaction>

<reaction text=”Re+D—Rb” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Re*D” />
</reaction>

<reaction text=”Re+D—Rp” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Re*D” />
</reaction>

<reaction text=”Rb+D—Rf” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Rb*D” />
</reaction>

<reaction text=”Rp+D—Rf” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Rp*D” />
</reaction>

<reaction text=”Rb—Re+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Rb” />
</reaction>

<reaction text=”Rp—Re+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Rp” />
</reaction>
```

```
<reaction text=”Rf—Rb+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Rf” />
</reaction>

<reaction text=”Rf—Rp+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Rf” />
</reaction>

<reaction text=”Ae+D—Ab” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Ae*D” />
</reaction>

<reaction text=”Ae+D—Ap” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Ae*D” />
</reaction>

<reaction text=”Ab+D—Af” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Ab*D” />
</reaction>

<reaction text=”Ap+D—Af” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_bind” species=”Ap*D” />
</reaction>

<reaction text=”Ab—Ae+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Ab” />
</reaction>
```

```
<reaction text=”Ap—Ae+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Ap” />
</reaction>

<reaction text=”Af—Ab+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Af” />
</reaction>

<reaction text=”Af—Ap+D” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_rdna” species=”Af” />
</reaction>

<reaction text=”Rf—Af” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”vmax” species=”Rf” />
</reaction>

<reaction text=”—Y” after=”—”>
        <delay type=”none” />
        <propensity type=”massaction” k=”k_Y” species=”A” />
</reaction>

<rule type=”additive” frequency=”repeated” equation=”A=Ae+Ab+Ap+Af” />
<rule type=”additive” frequency=”repeated” equation=”R=Re+Rb+Rp+Rf” />
<rule type=”additive” frequency=”repeated” equation=”I=If+D+D” />

<species name=”I_pl” value=”1” />

<species name=”Y” value=”0” />

<species name=”I” value=”0” />
<species name=”If” value=”0” />
```

```
<species name=”D” value=”0” />

<species name=”Re” value=”1” />
<species name=”Rb” value=”0” />
<species name=”Rp” value=”0” />
<species name=”Rf” value=”0” />
<species name=”R” value=”0” />

<species name=”Ae” value=”0” />
<species name=”Ab” value=”0” />
<species name=”Ap” value=”0” />
<species name=”Af” value=”0” />
<species name=”A” value=”0” />

<parameter name=”k_I” value=”0.83” />
<parameter name=”k_Y” value=”0.357” />
<parameter name=”k_bind” value=”1” />
<parameter name=”k_rdim” value=”30” />
<parameter name=”k_rdna” value=”30” />
<parameter name=”vmax” value=”0.01” />

</model>
```

Appendix B

A Mechanistic Model of Integrase Recombination is Not Identifiable

We used the integrase model in Appendix A.5 as a more mechanistically grounded model of integrase recombination dynamics. In this model, integrase molecules are produced at a rate k_I that is proportional to the amount of integrase plasmid I_{pl}. The integrase molecules can dimerize and undimerize with rates k_{bind} and k_{rdim} respectively. The dimers can bind and unbind to attP and attB sites with rates k_{bind} and k_{rdna} respectively. Note that the forward rates for binding are the same for both dimerization and binding to DNA. This forward rate is set to unity and only the reverse rate is identified. This is a way to improve identifiability of the model, because typically the identification of binding and unbinding rates is challenging. In this model, only reporter plasmids with dimers bound to both the attB and attP sites can recombine (with rate $vmax$) into activated reporter plasmids. The output Y (YFP) is produced at a rate k_Y and is proportional to the total concentration of activated reporter plasmids in the system. In this model, integrase molecules can still bind to the activated reporter plasmids; however, no further recombination occurs once a plasmid has become active. The full details of the model, including explicit reactions and parameter values, are contained in Appendix A.5.

First, we simulated the model using a hand-tuned set of parameters and found that the full mechanistic model could produce qualitatively similar results to the simpler Hill function model, as shown in Figure B.1. That is, increasing integrase plasmid concentration increased integrase expression and reduced delay in reporter activation. On the other hand, increasing reporter plasmid did not affect integrase expression but increased reporter expression.

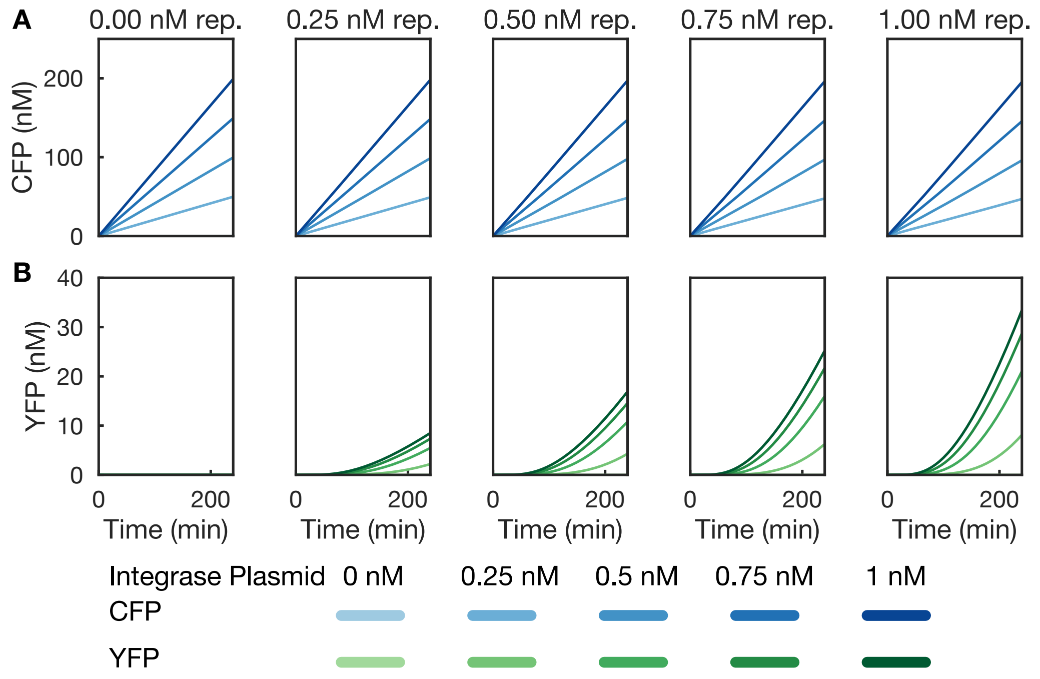


Figure B.1: Simulations of integrase (A) and reporter (B) expression using a full mechanistic model. Increasing integrase plasmid increases integrase (CFP) expression and decreases delay in reporter (YFP) activation. Increasing reporter plasmid does not affect integrase expression and increases reporter expression.

Then, using the same approach as with the simpler Hill function recombination model, we first assessed identifiability computationally by checking if an inference procedure could successfully recover the true model parameters from simulated data. We fixed the dimerization rate of integrases and the binding rate of integrase dimers to DNA both to unity, and we only identified the dissociation rates. In total, we attempted to identify the integrase throughput vmax, the dimer unbinding rate k_{rdim}, the DNA unbinding rate k_{rdna}, the integrase production rate k_I, and the reporter production rate k_Y.

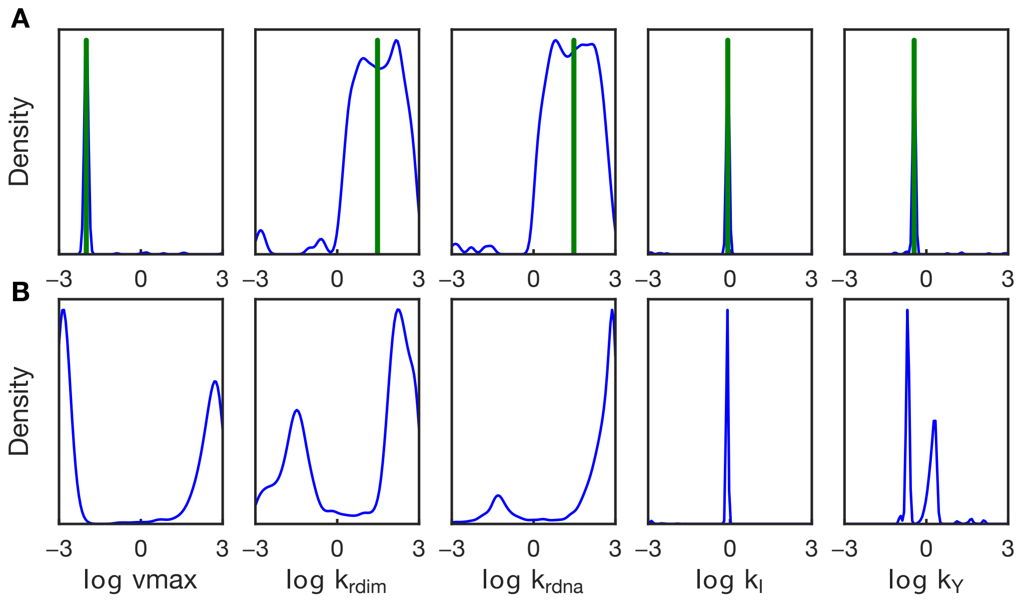


Figure B.2: Markov chain Monte Carlo shows that the full mechanistic model is only partially identifiable. (A) Computational identifiability analysis using simulated data shows that the posterior distributions (blue) only match the true parameters (green) for three of five parameters. (B) Parameter inference on the real data shows non-identifiability of three of five parameters.

As expected, and as seen in Figure B.2A, the unbinding parameters were much more difficult to identify than the recombination rate and production rates. In fact, while vmax, k_I, and k_Y were identified exactly, the unbinding parameters k_{rdim} and k_{rdna} were only narrowed down to three orders of magnitude (between 1 and 1000). There was a moderate negative correlation of $-.50$ between k_{rdim} and k_{rdna}. This is unsurprising because as the the integrase dimerizes more weakly, the dimers must bind more strongly to DNA to maintain the same rate of recombination.

Furthermore, when inference was performed on the real data (Figure B.2B), only the production rates k_I and k_Y had a tight posterior distribution. Thus, we believe that the simple Hill function model is preferable to the full mechanistic model for modeling integrase dynamics.

Bibliography

[1] *MATLAB and SimBiology Toolbox Release R2016a*. Natick, Massachusetts: The MathWorks Inc., 2016.

[2] B. D. O. Anderson, “The inverse problem of stationary covariance generation,” *Journal of Statistical Physics*, vol. 1, no. 1, pp. 133–147, 1969.

[3] J. C. Anderson, C. A. Voigt, and A. P. Arkin, “Environmental signal integration by a modular AND gate,” *Molecular Systems Biology*, vol. 3, no. 1, 2007.

[4] O. Banerjee, L. El Ghaoui, A. d’Aspremont, and G. Natsoulis, “Convex optimization techniques for fitting sparse Gaussian graphical models,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 89–96.

[5] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn, and K. Smith, “Cython: The best of both worlds,” *Computing in Science Engineering*, vol. 13, no. 2, pp. 31 –39, 2011.

[6] E. Billauer, “peakdet: Peak detection using matlab,” *http://www.billauer.co.il/peakdet.html*, 2012.

[7] J. Bonnet, P. Subsoontorn, and D. Endy, “Rewritable digital data storage in live cells via engineered control of recombination directionality.” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, no. 23, pp. 8884–8889, 2012.

[8] V. Brendel and A. S. Perelson, “Quantitative model of ColE1 plasmid copy number control,” *Journal of Molecular Biology*, vol. 229, no. 4, pp. 860–872, 1993.

[9] Y. Chen, J. K. Kim, A. J. Hirning, K. Josić, and M. R. Bennett, “Emergent genetic oscillations in a synthetic microbial consortium,” *Science*, vol. 349, no. 6251, pp. 986–989, 2015.

[10] M. W. Chevalier and H. El-Samad, “A data-integrated method for analyzing stochastic biochemical networks,” *The Journal of Chemical Physics*, vol. 135, no. 21, p. 214110, 2011.

[11] R. Daniel, J. R. Rubens, R. Sarpeshkar, and T. K. Lu, “Synthetic analog computation in living cells,” *Nature*, vol. 497, no. 7451, pp. 619–623, 2013.

[12] Y. Y. David Hayden and J. Goncalves, “Network reconstruction from intrinsic noise: Minimum-phase systems,” in *American Control Conference (ACC)*, 2014, pp. 4391–4396.

[13] D. Del Vecchio and R. M. Murray, *Biomolecular Feedback Systems*. Princeton University Press, 2014.

[14] D. Del Vecchio, A. J. Ninfa, and E. D. Sontag, “Modular cell biology: retroactivity and insulation.” *Molecular Systems Biology*, vol. 4, p. 161, 2008.

[15] A. P. Dempster, “Covariance selection,” *Biometrics*, vol. 28, no. 1, pp. 157–175, 1972.

[16] M. J. Dunlop, R. S. Cox III, J. H. Levine, R. M. Murray, and M. B. Elowitz, “Regulatory activity revealed by dynamic correlations in gene expression noise,” *Nature Genetics*, vol. 40, pp. 1493–1498, 2008.

[17] A. Edelstein, M. Tsuchida, N. Amodaj, H. Pinkard, R. Vale, and N. Stuurman, “Advanced methods of microscope control using μmanager software,” *Journal of Biological Methods*, vol. 1, no. 2, pp. 1–10, 2014.

[18] A. Eldar and M. B. Elowitz, “Functional roles for noise in genetic circuits,” *Nature*, vol. 467, no. 7312, pp. 167–173, 2010.

[19] M. B. Elowitz and S. Leibler, “A synthetic oscillatory network of transcriptional regulators,” *Nature*, vol. 403, no. 6767, pp. 335–338, 2000.

[20] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, “Stochastic gene expression in a single cell,” *Science*, vol. 297, no. 5584, pp. 1183–1186, 2002.

[21] C. Engler, R. Gruetzner, R. Kandzia, and S. Marillonnet, “Golden gate shuffling: A one-pot DNA shuffling method based on type IIs restriction enzymes,” *PLoS ONE*, vol. 4, no. 5, pp. 1–9, 2009.

[22] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, “emcee: The MCMC Hammer,” *arXiv*, vol. 125, p. 306, 2013.

[23] T. S. Gardner, C. R. Cantor, and J. J. Collins, “Construction of a genetic toggle switch in Escherichia coli,” *Nature*, vol. 403, no. 6767, pp. 339–342, 2000.

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

[25] K. Glover, “Structural aspects of system identification,” Ph.D. dissertation, Massachusetts Institute of Technology, 1973.

[26] A. Golightly and D. J. Wilkinson, “Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo,” *Interface Focus*, vol. 1, no. 6, pp. 807–820, 2011.

[27] M. M. Gomez, M. Sadeghpour, M. R. Bennett, G. Orosz, and R. M. Murray, “Stability of systems with stochastic delays and applications to genetic regulatory networks,” *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 4, pp. 1844–1873, 2016.

[28] J. Goodman and J. Weare, “Ensemble samplers with affine invariance,” *Communications in Applied Mathematics and Computational Science*, vol. 5, no. 1, pp. 65–80, 2010.

[29] A. C. Groth and M. P. Calos, “Phage integrases: Biology and applications,” *Journal of Molecular Biology*, vol. 335, no. 3, pp. 667 – 678, 2004.

[30] J. Hasenauer, C. Hasenauer, T. Hucho, and F. J. Theis, “ODE constrained mixture modelling: A method for unraveling subpopulation structures and dynamics,” *PLoS Computational Biology*, vol. 10, no. 7, pp. 1–17, 2014.

[31] J. Hasenauer, S. Waldherr, K. Wagner, and F. Allgöwer, “Parameter identification, experimental design and model falsification for biological network models using semidefinite programming,” *IET Systems Biology*, vol. 4, pp. 119–130(11), 2010.

[32] J. Hasenauer, S. Waldherr, M. Doszczak, N. Radde, P. Scheurich, and F. Allgöwer, “Identification of models of heterogeneous cell populations from population snapshot data,” *BMC Bioinformatics*, vol. 12, no. 1, p. 125, 2011.

[33] Y. Hori, T.-H. Kim, and S. Hara, “Existence criteria of periodic oscillations in cyclic gene regulatory networks,” *Automatica*, vol. 47, no. 6, pp. 1203 – 1209, 2011.

[34] Y. Hori and R. M. Murray, “A state-space realization approach to set identification of biochemical kinetic parameters,” in *2015 European Control Conference (ECC)*, 2015, pp. 2280–2285.

[35] V. Hsiao, E. L. C. de los Santos, W. R. Whitaker, J. E. Dueber, and R. M. Murray, “Design and implementation of a biomolecular concentration tracker.” *ACS Synthetic Biology*, vol. 4, no. 2, pp. 150–161, Feb. 2015.

[36] V. Hsiao, Y. Hori, P. W. Rothemund, and R. M. Murray, “A population-based temporal logic gate for timing and recording chemical events.” *Molecular Systems Biology*, vol. 12, no. 5, p. 869, 2016.

[37] V. Hsiao, A. Swaminathan, and R. M. Murray, “Control theory for synthetic biology,” *To appear, IEEE Control Systems Magazine*.

[38] T. Huang and J. G. Schneider, "Learning auto-regressive models from sequence and non-sequence data," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 1548–1556.

[39] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, *et al.*, "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, 2003.

[40] D. Huh and J. Paulsson, "Random partitioning of molecules at cell division," *Proceedings of the National Academy of Sciences*, vol. 108, no. 36, pp. 15 004–15 009, 2011.

[41] D. Husmeier, "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks," *Bioinformatics*, vol. 19, no. 17, pp. 2271–2282, 2003.

[42] F. Hussain, C. Gupta, A. J. Hirning, W. Ott, K. S. Matthews, K. Josić, and M. R. Bennett, "Engineered temperature compensation in a synthetic genetic clock," *Proceedings of the National Academy of Sciences*, vol. 111, no. 3, pp. 972–977, 2014.

[43] M. Jahn, C. Vorpahl, T. Hübschmann, H. Harms, and S. Müller, "Copy number variability of expression plasmids determined by cell sorting and droplet digital PCR," *Microbial Cell Factories*, vol. 15, no. 1, p. 211, 2016.

[44] T. Katayama, *Subspace Methods for System Identification*. Springer, 2005.

[45] J. D. Keasling, "Synthetic biology and the development of tools for metabolic engineering." *Metabolic Engineering*, vol. 14, no. 3, pp. 189–195, May 2012.

[46] M. Komorowski, B. Finkenstädt, C. V. Harper, and D. A. Rand, "Bayesian inference of biochemical kinetic parameters using the linear noise approximation," *BMC Bioinformatics*, vol. 10, no. 1, p. 343, 2009.

[47] J. Kuntz, M. Ottobre, G.-B. Stan, and M. Barahona, "Bounding stationary averages of polynomial diffusions via semidefinite programming," *SIAM Journal on Scientific Computing*, vol. 38, no. 6, pp. A3891–A3920, 2016.

[48] G. Lillacci and M. Khammash, "Parameter estimation and model selection in computational biology," *PLoS Computational Biology*, vol. 6, no. 3, pp. 1–17, 2010.

[49] G. Lillacci and M. Khammash, "The signal within the noise: efficient inference of stochastic gene regulation models using fluorescence histograms and stochastic simulations," *Bioinformatics*, vol. 29, no. 18, p. 2311, 2013.

[50] C. T. Lin, "Structural controllability," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 201–208, 1974.

[51] J. Lipinski-Kruszka, J. Stewart-Ornstein, M. W. Chevalier, and H. El-Samad, "Using dynamic noise propagation to infer causal regulatory relationships in biochemical networks," *ACS Synthetic Biology*, vol. 4, no. 3, pp. 258–264, 2015.

[52] T. K. Lu, A. S. Khalil, and J. J. Collins, "Next-generation synthetic gene networks." *Nature Biotechnology*, vol. 27, no. 12, pp. 1139–1150, 2009.

[53] E. Lubeck and L. Cai, "Single-cell systems biology by super-resolution imaging and combinatorial labeling," *Nature Methods*, vol. 9, no. 7, pp. 743–748, 2012.

[54] R. Lutz and H. Bujard, "Independent and tight regulation of transcriptional units in Escherichia Coli via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements," *Nucleic Acids Research*, vol. 25, no. 6, pp. 1203–1210, 1997.

[55] T. R. Maarleveld, B. G. Olivier, and F. J. Bruggeman, "Stochpy: A comprehensive, user-friendly tool for simulating stochastic biological processes," *PLoS ONE*, vol. 8, no. 11, pp. 1–10, 2013.

[56] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré, "Markov chain Monte Carlo without likelihoods," *Proceedings of the National Academy of Sciences*, vol. 100, no. 26, pp. 15 324–15 328, 2003.

[57] W. Mather, M. R. Bennett, J. Hasty, and L. S. Tsimring, "Delay-induced degrade-and-fire oscillations in small genetic circuits," *Physical Review Letters*, vol. 102, p. 068105, 2009.

[58] D. A. McQuarrie, "Stochastic approach to chemical kinetics," *Journal of Applied Probability*, vol. 4, no. 3, pp. 413–478, 1967.

[59] P. A. Meacock and S. N. Cohen, "Partitioning of bacterial plasmids during cell division: a cis-acting locus that accomplishes stable plasmid inheritance," *Cell*, vol. 20, no. 2, pp. 529 – 542, 1980.

[60] S. Meinhardt, M. W. Manley, N. A. Becker, J. A. Hessman, L. J. Maher, and L. Swint-Kruse, "Novel insights from hybrid LacI/GalR proteins: family-wide functional attributes and biologically significant variation in transcription repression," *Nucleic Acids Research*, vol. 40, no. 21, pp. 11 139–11 154, 2012.

[61] D. Mishra, P. M. Rivera, A. Lin, D. Del Vecchio, and R. Weiss, "A load driver device for engineering modularity in biological networks," *Nature Biotechnology*, vol. 32, no. 12, pp. 1268–1275, 2014.

[62] T. S. Moon, C. Lou, A. Tamsir, B. C. Stanton, and C. A. Voigt, "Genetic programs constructed from layered logic gates in single cells," *Nature*, vol. 491, no. 7423, pp. 249–253, Apr. 2013.

[63] S. J. Moore, J. T. MacDonald, S. Weinecke, N. Kylilis, K. M. Polizzi, R. Biedendieck, and P. S. Freemont, "Prototyping of Bacillus megaterium genetic elements through automated cell-free characterization and Bayesian modelling," *bioRxiv*, 2016, doi: *10.1101/071100*.

[64] B. Munsky and M. Khammash, "The finite state projection algorithm for the solution of the chemical master equation," *The Journal of Chemical Physics*, vol. 124, no. 4, p. 044104, 2006.

[65] B. Munsky and M. Khammash, "Identification from stochastic cell-to-cell variation: a genetic switch case study," *IET Systems Biology*, vol. 4, no. 6, pp. 356–366(10), 2010.

[66] B. Munsky, B. Trinh, and M. Khammash, "Listening to the noise: random fluctuations reveal gene network parameters," *Molecular Systems Biology*, vol. 5, no. 1, pp. 1–7, 2009.

[67] V. K. Mutalik, J. C. Guimaraes, G. Cambray, C. Lam, M. J. Christoffersen, Q.-A. Mai, A. B. Tran, M. Paull, J. D. Keasling, A. P. Arkin, and D. Endy, "Precise and reliable gene expression via standard transcription and translation initiation elements," *Nature Methods*, vol. 10, no. 4, pp. 354–360, 2013.

[68] G. Neuert, B. Munsky, R. Z. Tan, L. Teytelman, M. Khammash, and A. van Oudenaarden, "Systematic identification of signal-activated stochastic gene regulation," *Science*, vol. 339, no. 6119, pp. 584–587, 2013.

[69] H. Niederholtmeyer, Z. Z. Sun, Y. Hori, E. Yeung, A. Verpoorte, R. M. Murray, and S. J. Maerkl, "Rapid cell-free forward engineering of novel genetic ring oscillators," *eLife*, vol. 4, 2015.

[70] W. Pan, Y. Yuan, J. Goncalves, and G.-B. Stan, "A sparse Bayesian approach to the identification of nonlinear state-space systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 182–187, 2016.

[71] J. Paulsson, "Models of stochastic gene expression," *Physics of Life Reviews*, vol. 2, no. 2, pp. 157–175, 2005.

[72] J. Paulsson and M. Ehrenberg, "Noise in a minimal regulatory network: plasmid copy number control." *Q Rev Biophys*, vol. 34, no. 1, pp. 1–59, 2001.

[73] L. Potvin-Trottier, N. D. Lord, G. Vinnicombe, and J. Paulsson, "Synchronous long-term oscillations in a synthetic gene circuit," *Nature*, vol. 538, no. 7626, pp. 514–517, 2016.

[74] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman, "Population growth of human Y chromosomes: a study of Y chromosome microsatellites." *Molecular Biology and Evolution*, vol. 16, no. 12, p. 1791, 1999.

[75] A. Raj, P. van den Bogaard, S. A. Rifkin, A. van Oudenaarden, and S. Tyagi, “Imaging individual mRNA molecules using multiple singly labeled probes,” *Nature Methods*, vol. 5, no. 10, pp. 877–879, 2008.

[76] A. Rantzer, “On the Kalman-Yakubovich-Popov lemma,” *Systems & Control Letters*, vol. 28, no. 1, pp. 7–10, 1996.

[77] N. Roquet, A. P. Soleimany, A. C. Ferris, S. Aaronson, and T. K. Lu, “Synthetic recombinase-based state machines in living cells,” *Science*, vol. 353, no. 6297, 2016.

[78] J. Ruess, F. Parise, A. Milias-Argeitis, M. Khammash, and J. Lygeros, “Iterative experiment design guides the characterization of a light-inducible gene expression circuit.” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 112, no. 26, pp. 8148–8153, June 2015.

[79] K. Sachs, O. Perez, D. Pe’er, D. A. Lauffenburger, and G. P. Nolan, “Causal protein-signaling networks derived from multiparameter single-cell data,” *Science*, vol. 308, no. 5721, pp. 523–529, 2005.

[80] J. Shin and V. Noireaux, “An E. coli cell-free expression toolbox: Application to synthetic gene circuits and artificial cells,” *ACS Synthetic Biology*, vol. 1, no. 1, pp. 29–41, 2012.

[81] D. L. Shis, F. Hussain, S. Meinhardt, L. Swint-Kruse, and M. R. Bennett, “Modular, multi-input transcriptional logic gating with orthogonal LacI/GalR family chimeras,” *ACS Synthetic Biology*, vol. 3, no. 9, pp. 645–651, 2014.

[82] A. Singh and J. P. Hespanha, “Approximate moment dynamics for chemically reacting systems,” *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 414–418, 2011.

[83] M. C. M. Smith and H. M. Thorpe, “Diversity in the serine recombinases,” *Molecular Microbiology*, vol. 44, no. 2, pp. 299–307, 2002.

[84] F. St-Pierre, L. Cui, D. G. Priest, D. Endy, I. B. Dodd, and K. E. Shearwin, “One-step cloning and chromosomal integration of DNA,” *ACS Synthetic Biology*, vol. 2, no. 9, pp. 537–541, 2013.

[85] 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*, vol. 456, no. 7221, pp. 516–519, 2008.

[86] S. Stylianidou, C. Brennan, S. B. Nissen, N. J. Kuwada, and P. A. Wiggins, “Supersegger: robust image segmentation, analysis and lineage tracking of bacterial cells,” *Molecular Microbiology*, vol. 102, no. 4, pp. 690–700, 2016.

[87] A. Swaminathan, V. Hsiao, and R. M. Murray, "Quantitative modeling of integrase dynamics using a novel Python toolbox for parameter inference in synthetic biology," *bioRxiv*, 2017, doi: *10.1101/121152*.

[88] A. Swaminathan and R. M. Murray, "Linear system identifiability from distributional and time series data," in *American Control Conference (ACC)*, 2016, pp. 392–399.

[89] I. A. Swinburne, D. G. Miguez, D. Landgraf, and P. A. Silver, "Intron length increases oscillatory periods of gene expression in animal cells," *Genes & Development*, vol. 22, no. 17, pp. 2342–2346, 2008.

[90] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf, "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems," *Journal of The Royal Society Interface*, vol. 6, no. 31, pp. 187–202, 2009.

[91] J. P. Torella, F. Lienert, C. R. Boehm, J.-H. Chen, J. C. Way, and P. A. Silver, "Unique nucleotide sequence–guided assembly of repetitive DNA parts for synthetic biology applications," *Nature Protocols*, vol. 9, no. 9, pp. 2075–2089, 2014.

[92] N. G. van Kampen, "Stochastic processes in physics and chemistry," 1995.

[93] A. Veliz-Cuba, A. J. Hirning, A. A. Atanas, F. Hussain, F. Vancia, K. Josić, and M. R. Bennett, "Sources of variability in a synthetic gene oscillator," *PLoS Computational Biology*, vol. 11, no. 12, pp. 1–23, 2015.

[94] E. Yeung, J. Kim, J. Gonçalves, and R. M. Murray, "Global network identification from reconstructed dynamical structure subnetworks: Applications to biochemical reaction networks," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 881–888.

[95] J. W. Young, J. C. W. Locke, A. Altinok, N. Rosenfeld, T. Bacarian, P. S. Swain, E. Mjolsness, and M. B. Elowitz, "Measuring single-cell gene expression dynamics in bacteria using fluorescence time-lapse microscopy," *Nature Protocols*, vol. 7, no. 1, pp. 80–88, 2012.

[96] C. Zechner, J. Ruess, P. Krenn, S. Pelet, M. Peter, J. Lygeros, and H. Koeppl, "Moment-based inference predicts bimodality in transient gene expression," *Proceedings of the National Academy of Sciences*, vol. 109, no. 21, pp. 8340–8345, 2012.

[97] C. Zechner, M. Unger, S. Pelet, M. Peter, and H. Koeppl, "Scalable inference of heterogeneous reaction kinetics from pooled single-cell recordings," *Nature Methods*, vol. 11, no. 2, pp. 197–202, 2014.

[98] S. Zeng, S. Waldherr, and F. Allgöwer, "An inverse problem of tomographic type in population dynamics," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 1643–1648.

[99] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996.