Theoretical Investigation of Biological Networks Coupled via Bottlenecks in Enzymatic Processing

Curtis T. Ogle

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Physics

William Mather, Chair
Michel Pleimling
Patrick Huber
Shengfeng Cheng

May 11, 2016
Blacksburg, Virginia

Keywords: Post-translational Coupling, Toxin-antitoxin Modules, Enzymatic Degradation, Queueing Theory
Copyright 2016, Curtis T. Ogle
Theoretical Investigation of Biological Networks Coupled via Bottlenecks in Enzymatic Processing

Curtis T. Ogle

(ABSTRACT)

Cell biology is a branch of science with a seemingly infinite abundance of interesting phenomena which are essential to our understanding of life and which may potentially drive the development of technology that improves our lives. Among the open ended questions within the field, an understanding of how gene networks are affected by limited cellular components is both broad and rich with interest. Common to all cellular systems are enzymes which perform many tasks within cells without which organisms could not remain healthy. Here are presented several explorations of enzymatic processing as well as a tool constructed for this purpose. More specifically, these works consider the effect of coupling of gene networks via competition for enzymes found within the cell. It is shown that a limitation on the number of available enzymes permits the formation of bottlenecks which drastically affect molecular dynamics within cells. These effects potentially afford cell behaviors that in part explain the impressive robustness of life to constantly fluctuating environments.

This work received support from the National Science Foundation (NSF) “A Queueing Framework for Synthetic Circuits in E. coli,” #1330180
Acknowledgments

I would like to thank Dr. Will Mather for persistent guidance throughout my time as a member of his lab as well as Dr. Nicholas Butzin, both of which provided indispensable advice of both a personal and professional nature on countless occasions. I thank Dr. Phil Hochendoner for sharing in this experience with me as well. It was by his advice that I attempted to work within the Mather lab, a decision which has served me very well over the last few years. The authenticity of my teammates provided a healthy and constructive environment to explore new ideas of scientific nature or otherwise, and I am a better person for having worked with them. In the years that I have spent in Blacksburg as a student, for all of the help I have received without which I could not have continued, I am most thankful for the patience I was shown by so many faculty members, in particular those within the Math and Physics departments. There are frankly too many such faculty members to properly credit, but among those whom I feel showed me valuable aid and kindness when I was most in need, I must thank Dr. Leo Piilonen and Dr. Royce Zia. Most importantly, I must thank my family members, my father John Ogle, mother Kay Hollandsworth, and brother Brian Ogle, who provided a supreme environment in which to grow. Without their loving and tireless support, I could not have arrived at this point in my life. And to Charlotte Oliver, I am thankful for the love and emotional support that has helped to carry me throughout all of graduate school. I could not possibly be the person I am without these people, and I will never forget their immeasurable contributions to my life.
## Contents

1 Introduction ................................................. 1

2 Background .................................................. 2

   2.1 Molecular and Cell Biology .......................... 2

       2.1.1 Central Dogma .................................. 3

       2.1.2 Proteolytic Degradation ......................... 4

       2.1.3 Synthetic Biology ................................. 5

       2.1.4 Toxin-Antitoxin Modules ......................... 7

   2.2 Stochastic Modeling .................................... 8

       2.2.1 The Gillespie Algorithm ......................... 9

       2.2.2 High Performance Computation Using Python .... 11

   2.3 Queueing Theory ....................................... 12

       2.3.1 Queueing Regimes ................................ 13

       2.3.2 Applicability to Biology ....................... 13

3 Proteolytic Crosstalk in Multi-Protease Networks ........... 17

   3.1 Abstract .............................................. 17

   3.2 Introduction .......................................... 18

   3.3 Methods ................................................ 19

       3.3.1 Multi-Enzyme Crosstalk Model .................... 19

       3.3.2 Degrade-and-Fire Oscillator Model ............... 21

       3.3.3 Simulation Algorithm ............................ 22
3.3.4 Statistical Analysis .................................................. 22
3.3.5 Non-equilibrium Steady State Analysis of (2, 2) Queuing Network Behavior .................................................. 23

3.4 Results and Discussion ................................................. 24
3.4.1 Queueing Regimes for Multi-Enzyme Models .................. 24
3.4.2 Numerical Analysis of Queueing-Based Crosstalk for Simple Queuing Networks ................................................. 27
3.4.3 Numerical Results for Degrade-and-Fire (DF) Oscillators Coupled Through Queueing ................................................. 37

3.5 Conclusions ............................................................... 42
3.6 Acknowledgements ....................................................... 42

4 Proteolytically Coordinated Activation of Toxin-Antitoxin Modules 43
4.1 Abstract .......................................................................... 43
4.2 Introduction ...................................................................... 44
4.3 Methods .......................................................................... 46
4.3.1 Single TA Module Model ............................................. 47
4.3.2 Double TA Module Model ............................................ 49
4.3.3 Single TA Module Return Mechanisms .......................... 53
4.3.4 Simulation Algorithm .................................................. 54
4.3.5 Event Detection ........................................................ 54
4.3.6 Event Measurement .................................................... 56

4.4 Results and Discussion ................................................... 56
4.4.1 Single TA Module Model ............................................. 57
4.4.2 Double TA Module Model ............................................ 60
4.4.3 Return Mechanism I ................................................... 63
4.4.4 Return Mechanism II ................................................... 68

4.5 Conclusions ...................................................................... 73
4.6 Acknowledgements ....................................................... 74
List of Figures

2.1 Central dogma of molecular biology. The central dogma of molecular biology describes a basic picture of protein regulation. DNA is used in the process of transcription to produce mRNA. mRNA is used by ribosomes in the process of translation to form polypeptides. Polypeptides are folded into thermodynamically stable conformations often affording them their functionality and constituting proteins. Proteins sometimes bind to DNA applying some regulatory action on the process of transcription. Proteins are also degraded by cellular proteases providing cells with a mechanism to rapidly remove proteins. Though essential to the ability of cells to affect protein regulation, proteolytic degradation is sometimes ignored when presenting the central dogma of molecular biology. ................................................................. 4

2.2 Goodwin oscillator circuit. A minimal oscillator consisting of a delayed negative feedback loop is known as a Goodwin oscillator. The formation of functional repressor takes several steps which impart a delay between when its mRNA is transcribed and when the mature protein arrives in the system. Such a delay could arise because chaperones are required to fold the repressor, or perhaps because it does not repress without first forming a multimer. Mature repressor is degraded by proteases and acts as a negative regulator of the production of its mRNA. ................................................................. 7

2.3 Generic toxin-antitoxin module. A toxin-antitoxin module consists of two genes under one promoter. The antitoxin is encoded first followed by the toxin. Once produced, the toxin and antitoxin form a complex which neutralizes the activity of the toxin. Both the antitoxin and the complex it forms with the toxin act as negative regulators of the operon forming a feedback loop. The antitoxin is rapidly degraded by proteases within the cell, while the antitoxin of the complex can also be degraded, releasing the toxin into the cell where it can affect cell growth. Many toxins halt processes within the cell by inhibiting translation of specific mRNAs and thus modifying their associated protein concentrations. ................................................................. 8
2.4 Demonstration of queueing regimes. A measurement of the mean job count after some period of time as a function of the production parameter $\lambda$ demonstrates the three regimes of queueing theory. Measurement of the job count $X$ was made at 100 time units using ensembles of 10,000 realizations. $\omega$ and $\epsilon$ were fixed at 10 and 0.00001 respectively.

2.5 Demonstration of correlation resonance. Three distinct measurements of steady state correlation resonance taken using ensembles of 20,000 realizations are shown for a simple network with two protein classes competing for one protease class. $\lambda_1$ is varied between 1 and 50 with $\lambda_2$ fixed at 10 (green), 25 (blue), and 40 (red). $\omega$, $\epsilon$, and $\gamma$ were fixed at 50, 0.00001, and 0.01 respectively. For each curve, correlation resonance is identified with dashed black lines where $\lambda_1 + \lambda_2 = \omega$.

3.1 Schematic of simple crosstalk networks. A schematic of the queueing networks we consider in this work, labeled by $(M, N)$ to indicate $M$ substrate classes and $N$ enzyme classes. The $(1, 1)$ system represents a standard single class queueing model used in introductory queueing theory, while the remaining systems exhibit richer dynamics. In our model, the dynamical variables are the counts of substrate, and dimension of the dynamics is thus set by $M$. We do not explore the $(1, 2)$, $(1, 3)$, or $(2, 3)$ systems, since in our model, these do not appear to generate qualitatively new results beyond the depicted systems.

3.2 Results for a $(2, 1)$ queueing network. Results for a $(2, 1)$ queueing network (see Section 3.4.2 for details). (A) For this queueing network at balance, with equal sharing of protease ($\eta = 0.5$), the counts of two substrate molecules ($X_1$ and $X_2$) exhibit strongly correlated dynamics due to competition for enzyme processing. (B) Strongly correlated trajectories persist for unequal substrate affinities ($\eta = 0.2$). (C) The correlation coefficient between $X_1$ and $X_2$ as a function of $\eta$ and $\lambda_2$ shows that a strong correlation resonance appears very near the queueing theoretic balance point ($\lambda_1 + \lambda_2 = \omega$, dashed line) for a wide range of $\eta$ values. This result extends the investigation of correlation resonance observed for the case $\eta = 0.5$ [18, 71]. Parameters are $\lambda_1 = 15$, $\gamma = 0.0231$, $\omega = 30$, and $\epsilon = 10^{-5}$, with $\lambda_2$ and $\eta$ variable.
3.3 **Results for a (2, 2) queueing network.** Results for a (2, 2) queueing network (see Section 3.4.2 for details). (A) and (B) show the mean substrate counts for $X_2$ and $X_1$, respectively, as $\lambda_2$ is varied for different values of $\eta$. The balance point ($\lambda_1 + \lambda_2 = 2\omega$, dashed line) indicates the transition between underloaded and overloaded states, and it is observed that increasing the production rate of $X_2$ can strongly induce $X_1$ in the overloaded regime ($\lambda_1 + \lambda_2 > 2\omega$). Only the case $\eta = 0$ (no multi-enzyme crosstalk) failed to exhibit strong induction of $X_1$. (C) The correlation coefficient between $X_1$ and $X_2$ exhibits a strong correlation resonance at the balance point (dashed line) for a wide range of $\eta$ values. (D) Scanning $\eta$ on a log-scale, positive correlation is maintained for very small values of $\eta$. Only when $\eta < \epsilon$ is approximately satisfied does the correlation finally decrease. Parameters are $\lambda_1 = 10$, $\gamma = 0.0231$, $\omega = 10$, and $\epsilon = 10^{-5}$, with $\lambda_2$ and $\eta$ variable.

3.4 **Results for a (2, 2) queueing network with varying system-size.** Setting $\lambda_1 = \lambda_2 = \alpha$ and $\omega = \nu\alpha$, the steady state correlation coefficient between $X_1$ and $X_2$ was measured while scanning the coupling parameter $\eta$ and the system-size parameter $\alpha$. This measurement was performed for each combination of three values of $\nu$ and $\epsilon$. Measurement was over an ensemble of 20,000 simulation trajectories for each set of unique parameters. The parameter value $\gamma = 0.0231$ is kept constant across these simulations.

3.5 **Results for a (2, 2) queueing network.** A more detailed analysis of a particular (2, 2) system with small $\eta$ (small crosstalk) using the thermodynamic formalism of Schnakenberg [96] (see Section 3.4.2). We plot the steady state (A) probability, (B) cycle current, (C) cycle potential, and (D) cycle power (heat generation) for a system set to balance, with parameters $\lambda_1 = \lambda_2 = 10$, $\gamma = 0.0231$, $\omega = 10$, $\eta = 0.1$, and $\epsilon = 10^{-5}$. It is seen that the boundary states (either $X_1 = 0$ or $X_2 = 0$), which were associated with priority queueing behavior in Section 3.4.2, contribute strong cycle potentials that in turn generate non-equilibrium cycle currents. Similar cycle potentials and currents persist even for very small $\eta$, e.g. $\eta = 10^{-3}$. 

---

ix
3.6 Results for a (2, 2) queueing network with nonidentical enzymes. A queueing picture robustly predicts qualitative changes in statistics for a (2, 2) system when $\omega_1$ (the maximum reaction velocity of enzyme 1) is not necessarily equal to $\omega_2$ (the maximum reaction velocity of enzyme 2). The value of $\lambda_2$ was varied while fixing the value $\lambda_1 = 10$, thus transitioning the system through different queueing regimes. The values of $\omega_1$ and $\omega_2$ were independently varied $\pm 20\%$ from a central value of $\omega_1 = \omega_2 = 10.0$, leading to nine unique pairs of values for $(\omega_1, \omega_2)$. Both (A) the steady state mean substrate count of $X_1$ and (B) the steady state correlation between $X_1$ and $X_2$ exhibit qualitative changes when the system parameters cross the balance point condition $\lambda_1 + \lambda_2 = \omega_1 + \omega_2$. Namely, $X_1$ is induced in the overloaded regime $(\lambda_1 + \lambda_2 > \omega_1 + \omega_2)$, and the correlation exhibits a peak near the balance point. Each of the five possible balance point conditions for our choices of $(\omega_1, \omega_2)$ are represented by vertical dashed black lines. Measurement was over an ensemble of 20,000 simulation trajectories for each set of unique parameters. The parameter values $\gamma = 0.0231$, $\eta = 0.1$, and $\epsilon = 10^{-5}$ are kept constant across these simulations. The legend for Panels (A) and (B) is displayed to the right of these panels. Panels (C) and (D) are identical to (A) and (B), but the value $\epsilon = 1.0$ was instead used. Results in this case are qualitatively similar to the case $\epsilon = 10^{-5}$.

3.7 Results for a (3, 2) queueing network. Similar to Fig. 3.3, but for a (3, 2) queueing system (see Section 3.4.2 for details). Coupling between the substrates $X_1$ and $X_2$ is generated indirectly by a another substrate $X_3$. As in other cases, induction of mean values and a peak in correlation coefficient are closely related to the point of balance $(\lambda_1 + \lambda_2 + \lambda_3 = 2\omega$, black dashed line). An additional positive peak in correlation occurs for very small $\eta$ near the condition $\lambda_3 = 5$ (white dashed line). Parameters are $\lambda_1 = \lambda_2 = 10$, $\gamma = 0.0231$, $\omega = 15$, and $\epsilon = 10^{-5}$, with $\lambda_3$ and $\eta$ variable.

3.8 Results for a (2, 2) oscillator system. Results for a (2, 2) oscillator system (see Section 3.4.3 for details). Sample trajectories are presented for identical oscillators ($\nu = 1$) either with (A) $\eta = 0.0$ (no crosstalk) or (B) $\eta = 0.5$ (strong crosstalk). Oscillators with no crosstalk show no apparent synchronization, while oscillators with strong crosstalk show nearly perfect synchronization. (C) Further evidence of synchronization is revealed by measuring the steady state correlation between $X_1$ and $X_2$ for a wide range of oscillator parameters. Parameters are $\beta = 2$, $C_0 = 10$, $\alpha_1 = 100$, $\alpha_2 = \nu \alpha_1$, $\gamma = 0.0231$, $\omega = 10$, and $\epsilon = 10^{-5}$, with $\nu$ and $\eta$ variable.
3.9 **Illustration of queueing-based synchronization for a (2, 2) system.**
Illustration of queueing-based synchronization for a (2, 2) system. Shown are the mean values for a (2, 2) system as a function of time with initial conditions \(X_1 = 200\) and \(X_2 = 100\), for either (A) \(\eta = 0.0\), (B) \(\eta = 0.001\), (C) \(\eta = 0.1\), or (D) \(\eta = 0.5\). Even for relatively weak crosstalk, the mean values degrade to approximately the same point in time due to the effect of shared protease. If these components were instead oscillators, this effect can be imagined to synchronize the next firing time of the oscillators. Parameters are \(\gamma = 0.0231\), \(\omega = 10\), and \(\epsilon = 10^{-5}\).

3.10 **Results for a (3, 2) oscillator system.** Results for a (3, 2) oscillator system (see Section 3.4.3 for details). The steady state correlation between oscillator counts \(X_1\) and \(X_2\) is plotted on both (A) a linear scale and (B) a log scale for \(\eta\). Overall, the influence of a third substrate is observed to produce weak positive or negative correlations between the two oscillators. These correlations are sufficiently small that we claim synchronization is weak. Parameters are \(\beta = 2\), \(C_0 = 10\), \(\alpha_1 = 20\), \(\alpha_2 = 20\), \(\gamma = 0.0231\), \(\omega = 10\), and \(\epsilon = 10^{-5}\), with \(\lambda_3\) and \(\eta\) variable.

4.1 **Schematic of single TA module model based on mazEF in E. coli.** A single mRNA is produced encoding both a toxin and an antitoxin. The toxin forms a dimer, which then forms a complex with the antitoxin. This complex also forms a dimer. The antitoxin, complex, and complex dimer all repress transcription of the mRNA (red lines). Proteases can degrade free antitoxin and antitoxin bound in complexes (blue lines). Free toxin and its dimer inhibit translation of both the toxin and antitoxin by cleaving the mRNA encoding them (orange lines). All species are also subject to effective dilution by cell growth and division (not shown). The rate of dilution is dependent on the level of free toxin and its dimer because cell growth is slowed by the accumulation of translation inhibiting toxins.
4.2 Schematic of double TA module model with proteolytic coupling. With the addition of an identical mazEF-like module and a proteolytic pathway, the effect of proteolytic coupling can be measured. There are now four distinct proteolytic actions, where either pathway may process the antitoxin of either module. In the uncoupled case, the pathways shown by grey arrows would not be present. Maximal coupling requires that antitoxins do not prefer either proteolytic pathway over the other (both pathways effectively constitute a single pathway). The antitoxins are shown to repress their associated toxins because of the neutralization concomitant with complex formation (and autorepression of the operon). The toxin is shown to repress the antitoxin as it inhibits translation preventing the production of additional antitoxin (and toxin). With proteolytic coupling, the toxins of each module are coupled transitively (green arrows) because of antitoxin coupling via proteolysis. Although artificial, the effective separation of dilutive pathways presents a similar picture, except that toxin-toxin interaction is more direct as both toxins are diluted, as opposed to the strictly transitive coupling of toxins owed to proteolytic competition.

4.3 Event detection robustness. Event detection of total toxin ($T + T_2$) is demonstrated for the four corners of the parameter space explored with the single TA module model. The process of event detection requires identification of the high and low transition thresholds shown with dashed lines in each plot (see Section 4.3.5 for a detailed description of event detection). Filtered events are marked with red lines and square end points whose positions coincide with the transitions in and out of the toxic state. Event detection depends on three parameters ($z_0 = 0.2$, $w_0 = 0.125$, and $n_t = 5$) which robustly identified events for the considered parameter space. In addition to statistics of toxic events, the count of events found is also recorded (shown in the bottom plot for the single module model) to serve as a proxy for the quality of the measured statistics. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, and $\gamma = 1$, while $\alpha_t$ and $k_C$ are varied.
4.4 Measurement of a single TA module model. Event detection was performed on the total toxin \( (T + T_2) \) permitting characterization of the events generated by the model of a single mazEF-like module. Events were measured from a single long trajectory of length 1,000,000 time units for each distinct set of parameter values. For the range of parameters scanned, the lowest event count was more than 450 events (see Fig. 4.3). The bottom left plot demonstrates strong dependence of the steady state probability of being in the high toxin state and the rates of translation \((\alpha_t)\) and of complex formation \((k_C)\). The bottom right plot shows strong dependence of the mean total toxin count during high toxin events on the translation rate, while there is little dependence on the binding rate. Parameters are \(\sigma = 20, k_0 = 0.05, \nu = 1.25, \omega = 0.2, \beta_T = 5, \beta_g = 0.25, \mu = 100, \epsilon = 0.05, \) and \(\gamma = 1, \) while the translation rate \(\alpha_t\) and the binding rate \(k_C\) are scanned.

4.5 Characterization of single TA module toxic events. Using event detection and measurement of the total toxin \((T + T_2)\), the mean and standard deviation of both toxic event width and height are measured. Events were collected from a single long trajectory of length 1,000,000 time units for each distinct set of parameter values. For the range of parameters scanned, the lowest event count was more than 450 events (see Fig. 4.3). The top left plot shows strong dependence of mean toxic event width on the translation rate \(\alpha_t\) and relatively weak dependence on the rate of complex formation \(k_C\). The top right plot shows strong dependence of toxic event width standard deviation on \(\alpha_t\) and almost no dependence on \(k_C\). The bottom left plot shows strong dependence of mean toxic event height on \(\alpha_t\) and relatively weak dependence on \(k_C\). The bottom right plot shows weak dependence of toxic event height standard deviation on \(\alpha_t\) and \(k_C\). Parameters are \(\sigma = 20, k_0 = 0.05, \nu = 1.25, \omega = 0.2, \beta_T = 5, \beta_g = 0.25, \mu = 100, \epsilon = 0.05, \) and \(\gamma = 1, \) while \(\alpha_t\) and \(k_C\) are scanned.

4.6 Measurement of a double TA module model. Coupling of identical mazEF-like modules via shared proteolytic pathways and a global effect on growth rate is explored using the model described in Section 4.3.2. The event detection process is run on the total toxin count of each module \((T_1 + T_{12} + T_2 + T_{22})\), and additional statistics such as the correlation of toxins are calculated by considering the total toxin counts during the events of one module (see Section 4.3.6 for more details). Proteolytic coupling strongly dilates the width of toxic events and correlates the entry and exit of the toxic state for each module. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are \(\sigma = 20, k_0 = 0.05, \nu = 1.25, \omega = 0.2, \beta_T = 5, \beta_g = 0.25, \mu = 100, \epsilon = 0.05, \gamma = 1, \alpha_t = 200, \) and \(k_C = 1000, \) while \(\eta_p\) and \(\eta_g\) are varied.
4.7 Effect of coupling on toxic event probability and correlation. Coupling of identical mazEF-like modules via shared proteolytic pathways and a global effect on growth rate is explored using the model described in Section 4.3.2. Without either growth rate or proteolytic coupling, toxic events are effectively uncorrelated. Proteolytic coupling increases the steady state toxic state probability by dilating and correlating the toxic events of each module. Growth rate coupling lowers the toxic state probability while increasing toxin correlations. Both of these effects are observed for low values of the coupling parameters \( \eta_p \) and \( \eta_g \). In particular, there is little effect on toxic state probability or toxin count correlation during toxic events when \( \eta_p \) is increased above 0.1 (20% of the maximal extent of proteolytic coupling). It should be noted that identical TA modules contained in the same cell may only experience the condition \( \eta_g = 0.5 \). As the modules are identical, the top left and top right plots are interchangeable as are the bottom left and bottom right plots. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are \( \sigma = 20, k_0 = 0.05, \nu = 1.25, \omega = 0.2, \beta_T = 5, \beta_g = 0.25, \mu = 100, \epsilon = 0.05, \gamma = 1, \alpha_t = 200, \) and \( k_C = 1000 \), while \( \eta_p \) and \( \eta_g \) are varied.

4.8 Effect of coupling on toxic event width and height. Coupling of identical mazEF-like modules via shared proteolytic pathways and a global effect on growth rate is explored using the model described in Section 4.3.2. The mean (top left) and standard deviation (top right) of toxic event widths increases with proteolytic and growth rate coupling. The mean toxin count during toxic events (bottom left) increases when either form of coupling is present but does not strongly depend on the extent of the coupling. The standard deviation of toxin count during toxic events (bottom right) decreases when either form of coupling is increased, showing little dependence above a relatively low level of coupling (20% of the maximal extent of proteolytic coupling). Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are \( \sigma = 20, k_0 = 0.05, \nu = 1.25, \omega = 0.2, \beta_T = 5, \beta_g = 0.25, \mu = 100, \epsilon = 0.05, \gamma = 1, \alpha_t = 200, \) and \( k_C = 1000 \), while \( \eta_p \) and \( \eta_g \) are varied.
4.9 Effect of return mechanism I on toxic events. Return mechanism I involves three parameters. $\rho_X$ and $\alpha_X$ are the transcription and translation rates of the additional species $X$, and $\beta_X$ regulates the affinity of the protease for the toxin and its dimer. The top left plot shows that with fast production of the species $X$ ($\rho_X = 1.0$ and $\alpha_X = 1.0$) the proposed return mechanism can strongly reduce the duration of toxic events. The bottom left plot shows that with far lower $\beta_X$, events are no longer stifled and the accumulation of $X$ far outpaces that of the toxin. With far lower production parameter values, the top right and bottom right plots show that $\beta_X$ can regulate the width of toxic events with relatively low accumulation of the additional species $X$. The count of events used for statistics are also shown for several values of $\beta_X$, where the minimum event count measured was 160. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.

4.10 Effect of return mechanism I on event probability. A measurement of steady state toxic state probability as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.

4.11 Effect of return mechanism I on event width. A measurement of the mean toxic event width as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.

4.12 Effect of return mechanism I on event height. A measurement of the mean toxin count during toxic events as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
4.13 **Effect of return mechanism II on toxic events.** Return mechanism II involves three parameters. $\rho_X$ and $\alpha_X$ are the transcription and translation rates of the additional species $X$, and $\beta_X$ regulates the affinity of the protease for the antitoxin bound to toxin in complexes. The top left plot shows that with fast production of the species $X$ ($\rho_X = 1.0$ and $\alpha_X = 1.0$) the proposed return mechanism can strongly reduce the duration of toxic events. The bottom left plot shows that with far lower $\beta_X$, events are no longer stifled and the accumulation of $X$ far outpaces that of the toxin. With far lower production parameter values, the top right and bottom right plots show that $\beta_X$ can regulate the width of toxic events with relatively low accumulation of the additional species $X$. The count of events used for statistics are also shown for several values of $\beta_X$, where the minimum event count measured was 160. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.

4.14 **Effect of return mechanism II on event probability.** A measurement of steady state toxic state probability as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.

4.15 **Effect of return mechanism II on event width.** A measurement of the mean toxic event width as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.

4.16 **Effect of return mechanism II on event height.** A measurement of the mean toxin count during toxic events as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
5.1 **Simulated annealing result for simple enzymatic system.** Using 50 processes on a laptop with four cores, the above fit was found in fewer than two minutes using a series of discrete and continuous fitting iterations. The simple network used has three reaction rates (binding rate, unbinding rate, and enzymatic processing rate) which were used as axes of the parameter space over which to perform parameter estimation. Each axis was bounded below by $10^{-7}$ and above by $10^{6}$ and set at an initial value of 1, emulating the situation where essentially nothing is known about the correct values of the parameters. The input data set used was a single realization using the parameter values 1, 0.01, and 800 for the binding, unbinding, and processing rates respectively. The parameters found to have the best fit were 0.9766, 0.0001, and 786.6.

5.2 **Plotting interface of Modular.** The plotting interface of Modular consists of three major regions outlined in the above with three distinct colors. The green box outlines the resulting plot based on selections from the other two boxes. A toolbar below the plot provides the ability to pan, zoom, and save the image. The blue box contains various options for configuring the plot. Many of the controls are not visible in the above image, but this box contains controls for the axis labels, bounds, whether or not they are in log-space, which lines are visible, their widths, colors, symbols, and labels, and more. The red box contains a tree that associates a set of trajectories (only one trajectory in the provided image) with each distinct set of parameters which was considered in the run. For measurements which act on the entire parameter space, there is simply a set of trajectories unassociated with any particular position in parameter space.

5.3 **Correlation resonance for a simple proteolytic network.** Correlation resonance is shown for several parameter spaces explored by the three dimensional parameter scan described in the above example mcfg file. This measurement required over 8.2 million realizations of the Gillespie algorithm and was performed on two laptops (6 processing cores total using MPI) in fewer than 100 minutes.
Chapter 1

Introduction

Cell biology is a branch of science with a seemingly infinite abundance of interesting phenomena which are essential to our understanding of life and which may potentially drive the development of technology that improves our lives. Among the open ended questions within the field, an understanding of how gene networks are affected by limited cellular components is both broad and rich with interest. Common to all cellular systems are enzymes which perform many tasks within cells without which organisms could not remain healthy. It is shown that a limitation on the number of available enzymes permits the formation of bottlenecks which drastically affect molecular dynamics within cells. These effects potentially afford cell behaviors that in part explain the impressive robustness of life to constantly fluctuating environments. The following chapters discuss topics largely in the context of molecular and cell biology. All of the results presented will be theoretical, arising from computational simulations. However, this work was conducted tangentially to experiments run by the Mather lab, and it attempts to provide predictive understanding to open questions about cellular processing. It is nonetheless hoped that some of this material may be useful to others who are not studying cell biology, but perhaps other fields of science or engineering which require computational simulation or perhaps chemical networks outside of the context of biology.

Chapter 2 includes a basic primer on cell biology, computational methods for chemical networks, and queueing theory, which constitute several unifying themes in the following chapters. Chapter 3 presents published work exploring the subject of cellular processing theoretically using software developed over several years within the Mather lab. In Chapter 4, the intuition distilled from the work presented in Chapter 3 is used to explore the increasingly important class of gene network known as a “toxin-antitoxin module”. The software used to perform the computational simulations as seen throughout Chapters 3 and 4 is presented in Chapter 5. As well as for simulations, older versions of that software were used to construct additional software which was necessary to conduct many experiments within the Mather lab. These additional utilities are briefly discussed in Chapter 6. Closing remarks are provided in Chapter 7.
Chapter 2

Background

The purpose of this chapter is primarily to provide supporting information about the fields of molecular and cell biology and biophysical modeling. It is in no way exhaustive and aims strictly for a foundation of information sufficient to understand the following chapters. To this end there is supporting information about stochastic simulation of chemical networks and queueing theory, and their application to cell biology. There is also some discussion of several experimental techniques which help to bridge the gap between theoretical understanding and experimental results. The following chapters will not focus as heavily on experimental work and thus this chapter is aimed primarily at theoretical techniques and their applications to cell biology.

2.1 Molecular and Cell Biology

Cell biology considers systems comprised of cells such as bacterial colonies and the tissues which compose multicellular organisms. Molecular biology considers the dynamics of the contents of cells responsible for cellular functions. Cells are membrane bound subunits with genetic material and metabolic activity. They consume content from the environment, excrete byproducts, grow or shrink in size, and divide to produce progeny. They can interact with the environment by producing chemical signals in response to different stimuli, in some cases acting collectively by coupling to a shared chemical signal. Some have structures on their outer surface used to exert physical forces, perhaps providing propulsion towards food or adhesion to a stable surface [1,101]. Within cell biology, a system is characterized by the state of the environment and the cells which live there.
2.1.1 Central Dogma

The central dogma of molecular biology consists of several components and processes which describe a basic picture of protein synthesis (see Figure 2.1), the understanding of which is sufficient to describe many biological systems composed of gene networks [101]. Gene networks reside in cells, which can compose the tissues of multicellular organisms or exist in seeming singularity within an incredible range of environments. The range of environments in which a cell can survive and the manner in which a cell copes with its environment are generally functions of the genes which that cell possesses. Genes are encoded on deoxyribonucleic acid (DNA) as sequences of nucleotides. A nucleotide is a cellular molecule consisting of a base chemical (adenine, thymine, guanine, or cytosine) bound to a sugar and a phosphate group. The nucleotides are bound together forming strands. DNA is composed of two strands of nucleotides, bound in such a way that either strand may encode a distinct array of genes, serving as blueprints for many of the components of life which are not sequestered from the environment by cellular organisms [1,101].

The cellular component RNA polymerase binds to specific regions of DNA called “promoters”, which allows the initiation of a process called “transcription”. In transcription, DNA is used as a template to form ribonucleic acid (RNA). RNA is a single stranded sequence composed of the same nucleotides as DNA except with uracil in lieu of thymine. RNA has many functions within the cell beyond the passive encoding of other molecules as genes. For instance, some RNA sequences can bind to DNA, possibly related to their own production or the production of other distinct RNA sequences. Bound to the DNA, these RNA sequences modify the activity of RNA polymerase, resulting in regulation of the production of RNA sequences. A molecule, most often a protein, which binds to DNA resulting in regulation of RNA production is known as a “transcription factor”. Transcription factors may promote or hinder RNA polymerase in transcribing specific RNA sequences. RNA polymerase is an example of an important class of cellular component called an “enzyme.” Enzymes are proteins that catalyze reactions at a far greater rate than they would naturally occur [1,101].

mRNA (messenger RNA) is a type of RNA which can act as a template in the formation of polypeptides (chains of amino acids) using ribosomes in a process called “translation”. Amino acids are sometimes produced by the cell and sometimes sequestered from the environment. These polypeptides can fold into stable shapes forming proteins. Proteins are responsible for many necessary functions within the cell, while the function of a protein is largely dictated by the manner in which it folds. The folding of a protein is mostly dependent on the particular sequence of amino acids which compose the protein and generally determines its final shape. Some proteins function as transcription factors, binding to DNA and regulating the production of various RNA sequences [1,101]. The activity of a cellular organism is largely characterized by the concentrations of its proteins.

Transcription and translation are both processes which result in the production of necessary cellular molecules. It is also necessary for cells to be capable of removing molecules, such as dysfunctional unfolded or misfolded proteins or damaged RNA molecules [1,33,101].
Enzymes known as “proteases” are responsible for degrading proteins and misfolded polypeptides in a process known as “proteolysis” [1, 16, 33]. Proteolysis involves the unfolding of a protein and the breaking of its polypeptide bonds, allowing the amino acids composing the protein to be reused in the formation of other proteins [1, 33, 101]. Healthy cells in a bacterial colony typically increase in size over time, periodically dividing into multiple cells, resulting in a division of the molecules found within the membrane of the cell prior to division. This provides another mechanism to reduce the concentration of cellular molecules in a manner similar to and also termed “dilution” [71, 80]. The activity of a cellular organism is largely characterized by the concentrations of its proteins, while the processes of transcription, translation, proteolysis, and cell growth and division provide common central mechanisms to regulate these concentrations.

Figure 2.1: Central dogma of molecular biology. The central dogma of molecular biology describes a basic picture of protein regulation. DNA is used in the process of transcription to produce mRNA. mRNA is used by ribosomes in the process of translation to form polypeptides. Polypeptides are folded into thermodynamically stable conformations often affording them their functionality and constituting proteins. Proteins sometimes bind to DNA applying some regulatory action on the process of transcription. Proteins are also degraded by cellular proteases providing cells with a mechanism to rapidly remove proteins. Though essential to the ability of cells to affect protein regulation, proteolytic degradation is sometimes ignored when presenting the central dogma of molecular biology.

2.1.2 Proteolytic Degradation

Degradation is a process of particular necessity for protein regulation [16, 33] and consequently of interest throughout the following chapters. Many proteases work in concert with chaperones. Chaperones more generally function to fold proteins, refold misfolded proteins,
unfold proteins for degradation by a protease [1, 43, 101], and can function as adaptors to direct proteins (e.g. adaptor protein SspB is a chaperone that functions to associate specific proteins with the protease ClpXP) [16, 23, 50, 73]. Proteases such as ClpXP and ClpAP are composed of a chaperone subunit responsible for unfolding proteins (“foldase”) and a subunit which breaks the polypeptide bonds of an unfolded protein (“peptidase”), releasing the amino acids for reuse. ClpA and ClpX are chaperone subunits which unfold specific proteins. The peptidase subunit ClpP cannot easily degrade targets that an associated chaperone has not unfolded, and thus the chaperone subunit effectively regulates the affinity of the ClpAP and ClpXP proteases for particular proteins [16, 33, 43].

Adaptor proteins such as SspB are capable of recognizing specific molecular tags and can sometimes bind to targets, bringing them towards the protease ClpXP and tethering them to the ClpX chaperone subunit for eventual processing [16, 50, 73]. As the peptidase subunit may be occupied when the protease encounters a target, adaptor proteins which associate substrates with chaperone subunits allow the protease to generate a queue of targets to be degraded as soon as the peptidase finishes its current target. This enhances the efficiency of the protease by helping prevent the peptidase from being unoccupied and thus idle [50]. Adaptor proteins can thus provide an additional layer of control over the specificity of proteolysis within the cell, as the affinity of a substrate may effectively be regulated by the amount of adaptor proteins in the cell.

In addition to adaptor proteins, cells sometimes use short peptide sequences to mark a particular protein as a target for degradation. These sequences are recognized by chaperones such as ClpX or SspB. For example, translation may require amino acids which are unavailable to the cell or mRNA may be damaged or lacking a terminating signal for the ribosome (“stop codon”). These situations can cause ribosome stalling which promotes faulty or incomplete proteins. Cells require mechanisms to affect quality control on their protein products. For example, in *Escherichia coli* (*E. coli*) these faulty products are sometimes marked with an 11 amino acid tag known as SsrA [33, 43, 48]. Peptide tags are yet another effective layer of control over the specificity of proteolysis.

### 2.1.3 Synthetic Biology

Synthetic biology is a subfield of biology at the intersection of cell biology and engineering. It combines lab techniques with analytic techniques to engineer organisms with specific genes. These modifications can lead to stable and controllable dynamics and are often considered as discrete subunits, or “synthetic circuits.” Combinations of such circuits can be used to engineer cell populations which achieve some goal, such as emitting oscillatory light in the presence of a particular chemical or targeting a protein to a particular protease [9, 73, 85, 87, 88, 100].

In addition to engineering applications with commercializable goals [70, 84–86], synthetic biology provides a powerful tool for understanding nonsynthetic systems found in nature [9,
There are many synthetic approaches including removing or adding additional copies of genes, effectively modifying the production rate of their associated RNA molecules. This can be used to probe the importance of a gene in achieving some cellular behavior, such as the persistent response described previously [68]. A common technique is to insert into the DNA a nucleotide sequence which encodes a fluorescent protein tag. The fluorescent tag is a molecule bound to the protein which emits light near a specific wavelength. Measuring changes in the intensity of this light provides a proxy measurement of changes in the protein concentration and can be performed using modern microscopy techniques [9, 20, 77, 82, 84, 85, 100, 107].

Of particular interest throughout cell biology is the rod-shaped bacteria *E. coli*. *E. coli* is found in the intestines of humans, divides relatively quickly under healthy conditions, and creates more than 4,000 distinct proteins. It is one of the most well understood organisms in biology, and is consequently a particularly attractive target for synthetic approaches. As a model organism, it has provided a basis of knowledge applicable to many other organisms including humans [1]. There are several well understood synthetic circuits developed for use within *E. coli* [20, 35, 77, 84, 87, 100]. Because of this, it will often be a source of intuition when constructing theoretical models in later chapters.

Among the functional synthetic circuits which have been engineered, synthetic oscillator circuits are among the most commonly studied [77, 80, 84, 87, 100]. They have historically been an attractive topic for their relevance in real biological systems [9, 80]. Synthetic oscillators typically require proteolysis to create stable oscillations of the concentration of a protein which is usually modified to contain a fluorescent tag for measurement [9, 69, 77, 80, 84, 100]. Another common feature of synthetic oscillators are positive and negative feedback loops achieved with transcriptional regulation [9, 69, 87, 100]. In some cases this provides a means of tuning the oscillations using inducers [85, 100]. In later chapters, oscillating networks will be considered theoretically in part because of their experimental approachability using synthetic biological techniques and for their dependence on proteolysis. Figure 2.2 contains a diagram of a minimal oscillator circuit known as a Goodwin oscillator which only requires a delay in the production of an autorepressing protein [87, 94].

A uniquely suitable technique for determining the single cell behavior of synthetic (and natural) circuits is microfluidics. A microfluidic chip permits the aggregation of small colonies of bacteria which can be forced into a monolayer using microfluidic traps. Traps can be imaged using fluorescent microscopy techniques, providing time course measurements of protein concentrations for a colony of cells. The chip can provide channels which allow for the nearly uniform introduction of chemicals which interact with the cells. Some such chemicals, known as inducers, provide a regulatory effect on specific genes by acting as transcription factors. The addition of these chemicals can be modulated quickly, allowing for fast and uniform signaling to the colony. Microfluidics provides data which can be distilled and compared to results of stochastic simulations, providing an important intersection between experiment and theory that potentially validates models [9, 20, 77, 84, 85, 100].
Figure 2.2: **Goodwin oscillator circuit.** A minimal oscillator consisting of a delayed negative feedback loop is known as a Goodwin oscillator. The formation of functional repressor takes several steps which impart a delay between when its mRNA is transcribed and when the mature protein arrives in the system. Such a delay could arise because chaperones are required to fold the repressor, or perhaps because it does not repress without first forming a multimer. Mature repressor is degraded by proteases and acts as a negative regulator of the production of its mRNA.

### 2.1.4 Toxin-Antitoxin Modules

All bacteria exhibit a behavior known as persistence, where cells enter an almost dormant state characterized by very little metabolic activity and no apparent growth or division. These “persister” cells also achieve robustness against many stressors allowing them to survive events which kill the non-dormant cells of the population. The phenomenon of bacterial persistence was observed over 60 years ago and has since been an ongoing topic of study in Biology. The persistent state can be entered stochastically because of environmental fluctuations or because of some acute stress such as antibiotics or starvation [5,31,37,66–68]. The fraction of the population which occupies a persistent state depends on the environmental stress and maturity of the bacterial population but is generally very small. The persistence phenomenon is particularly relevant in consideration of the robustness of biofilms [31,37,66,67]. Biofilms are mature bacterial populations which collectively behave similarly to a multicellular organism, and are often capable of surviving antibiotic dosing and stress such as starvation [24]. The persistence phenomenon is largely responsible for chronic infections in humans, making it an attractive topic within the fields of biology and medicine. Recent evidence supports the necessity of toxin-antitoxin (TA) modules in promoting bacterial persistence [12,13,31,37,66–68].
A TA module consists of two or more genes arranged in an operon (a single mRNA is transcribed encoding the product of each gene [101]), at least one encoding a toxin and another encoding a cognate antitoxin. The toxin exerts some deleterious effect on the cell which slows or halts growth while the antitoxin interacts with the toxin to neutralize its effect. The toxin is relatively stable compared to the antitoxin which is readily degraded by proteases. The toxin and antitoxin proteins of a module can form complexes. The antitoxin and the complexes it forms are often capable of regulating the operon which produces the toxin and antitoxin mRNAs [12,13,37,42,66,68,110], constituting a feedback loop within the gene network which is common among TA modules. Chapter 4 discusses stochastic modeling of networks with one or two TA modules.

Figure 2.3: **Generic toxin-antitoxin module.** A toxin-antitoxin module consists of two genes under one promoter. The antitoxin is encoded first followed by the toxin. Once produced, the toxin and antitoxin form a complex which neutralizes the activity of the toxin. Both the antitoxin and the complex it forms with the toxin act as negative regulators of the operon forming a feedback loop. The antitoxin is rapidly degraded by proteases within the cell, while the antitoxin of the complex can also be degraded, releasing the toxin into the cell where it can affect cell growth. Many toxins halt processes within the cell by inhibiting translation of specific mRNAs and thus modifying their associated protein concentrations.

### 2.2 Stochastic Modeling

Stochastic models have proven effective at describing cellular systems in part because of their discrete and noise dependent nature [7, 9, 26, 27, 53, 63, 69, 81, 91, 104]. Cells are small volumes where molecular counts can be very low. It is particularly common to simulate just
Chapter 2. Background

the activity of a single cell at a time, extrapolating statistics of a cell population from those of an ensemble of independent single cell simulations. The cell is often treated as a fixed volume containing a solution of molecules such that simulation methods typically used for chemical reaction networks apply [9, 69, 77]. This is not completely correct in that cells can permit the flux of some molecules across their membranes, and variations in this flux across the cells will not generally be represented. Microfluidic experiments provide a relevant appropriate context for this assumption. Cells also inherit their initial state from their parent cell, which is potentially difficult to represent computationally. Deterministic simulations are sometimes granted an element of randomness by sampling distributions for initial conditions [77]; however, these effects will generally be ignored in later chapters when only steady state statistics are required as stochasticity of the simulations will obscure unrealistic initial conditions.

2.2.1 The Gillespie Algorithm

The Gillespie algorithm commonly used to simulate networks of chemical reactions is appropriate for cellular systems [29, 39, 57, 64, 80, 100, 104]. Given a set of elementary chemical reactions, the algorithm provides a sequence of steps for exact simulation of a well-mixed chemical solution [11, 39, 40, 104]. An elementary reaction consists of two or fewer reagents, possibly a set of products, and a function related to the rate at which the reaction occurs (reactions with three or more reagents are possible, but real networks can in general be decomposed into second or lower order reactions) [38]. The set of species in the system is effectively defined by the set of reagents and products used by all reactions in the network. The quantity of each chemical species is held as an integer molecular count, and each reaction modifies its reagents and products by integer amounts. Molecular counts can be related to concentrations by imposing a volume on the system, though this can present problems with time-dependent reaction rates [29, 39, 104].

The algorithm requires two particular calculations. Given the state of the system, a quantity known as a propensity is determined for each reaction. A propensity is related to the number of combinations of molecules in the system which are capable of interacting to perform the reaction [29, 39], and is consequently proportional the amount of each reagent currently in the system. By dividing the propensity of each reaction by the sum of all of the propensities ($\alpha_{\text{total}}$), the probability of each reaction to be the next to occur is calculated. Using a random number sampled from the interval $(0, 1)$, these probabilities are used to select the next reaction. As a second-order reaction may require two of the same or distinct species, the four example reactions below along with their associated propensity functions demonstrate all cases required for a basic implementation of the algorithm [29]:
Chapter 2. Background

\[ \varnothing \xrightarrow{k_1} Z \quad \alpha_1 = k_1 \quad (2.1) \]

\[ X \xrightarrow{k_2} Z \quad \alpha_2 = Xk_2 \quad (2.2) \]

\[ X + X \xrightarrow{k_3} Z \quad \alpha_3 = \frac{X(X - 1)}{2}k_3 \quad (2.3) \]

\[ X + Y \xrightarrow{k_4} Z \quad \alpha_4 = XYk_4 \quad (2.4) \]

\[ \alpha_{\text{total}} = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \quad (2.5) \]

In addition to the propensity of each reaction, the amount of time until the next reaction must be determined. Again by uniformly sampling the interval \((0, 1)\) for a random number \(r\), the following formula yields the time until the next reaction event \([29,39]\):

\[ \Delta t = -\frac{\ln(r)}{\alpha_{\text{total}}} \quad (2.6) \]

This yields a sequence of reaction events with exponentially distributed periods of time between events during which the system does not change state. A special case which must be accounted for occurs when \(\alpha_{\text{total}}\) is zero, in which case the time until the next reaction is undefined. This happens when no reaction is possible, and thus a fake reaction can be performed which requires the update of all reaction propensities despite having no effect on the molecular counts. A maximum time step must be provided for use when this occurs. For some systems this state can be inescapable; however, if either time dependent rates or zeroth-order reactions have been used, the system can recover from this state, and thus in general the simulation cannot be aborted even if all species counts reach zero.

There are many variations of the algorithm often intended for optimization in a particular context, sometimes compromising accuracy of the simulation \([11,38,40]\). Below is a short summary of one realization of the algorithm used throughout the following chapters, which is purely stochastic and avoids propensity calculations by using a variation of the next reaction method \([38]\). The implementation is appropriate for low molecular counts and high noise, and does not compromise accuracy to achieve optimization.

- The initial count of each species is set, the current time is set to zero, and the initial propensity of each reaction is calculated and entered into a data structure.

- The main simulation loop, which consists of four primary tasks, executes until some criterion is met, typically stopping when the current time is above a certain value.
  - The total propensity \((\alpha_{\text{total}})\) for all reactions is calculated.
– If $\alpha_{total}$ is zero, the next reaction will be the fake reaction which does not change the system, and the time until that reaction will be the maximum allowed time step. Otherwise, two independent random numbers are chosen on the interval $(0, 1)$ and used to decide the next reaction and the time until that reaction.

– The state of the system is potentially recorded, depending on a rule which forces the resulting data to represent a time domain with evenly spaced points.

– The current time is updated. The chosen reaction is performed which updates the current molecular counts. Reactions whose propensities are affected by the change in the system are updated, following the next reaction method. Propensities which depend on time-dependent rates are updated regardless of the reaction chosen.

• The resulting data consists of molecular counts as a function of time.

While the Gillespie algorithm described here only approximately represents time dependent reaction rates, modifications of the algorithm have been developed which are exact in these cases [64]. Time dependent functional forms can be used to represent external signals experienced uniformly by each cell in a population [9].

Using functional forms for reaction rates allows for simple representations of the processes of the central dogma (see Figure 2.1). Dynamics of transcription are sometimes encapsulated in a birth reaction whose rate has a functional form which depends on various molecular species of the system, representing transcriptional regulation [9,69,80] (see chapters 3 and 4 for examples.) For the process of translation, RNA molecules and ribosomes may or may not be explicitly simulated depending on the context and required measurements. The processes of both transcription and translation may be represented using zeroth, first, or second-order reactions depending on the effects being examined (in later chapters both transcription and translation will be represented using zeroth or first-order reactions.) Proteolysis and dilution by way of cell growth and division are post-translational processes. Dilution is typically represented as a first-order death reaction [18,71] with a rate constant related to the distribution of time between cell division events. Proteolysis is sometimes modeled using two reactions by introducing an intermediate species to represent the protease bound to the substrate [71,80]. In later chapters, a first-order reaction will be used to represent proteolysis with an appropriate functional form encapsulating specificity for each target and other known qualities of enzymatic processing (e.g. saturation) [80].

### 2.2.2 High Performance Computation Using Python

Almost all of the data presented in later chapters was generated using the Python Programming Language [93]. This was an active decision intended to leverage the ease of use and extendibility of Python. Despite the immediate speed gap between Python and typical languages used in science such as C++, with particular effort Python proved to be very capable.
There are existing options when looking for software capable of faithfully simulating biological chemical networks [65, 79, 90, 99, 106]. This remains true when only stochastic simulations are desired, and again when only Python-accessible software is considered [65, 106]. Despite this, it is not uncommon for computational biologists to implement their own simulation algorithms, often providing very specific statistical measurements and optimizations relevant to their systems of interest [9, 50, 69, 100]. Throughout later chapters, data will be shown which was generated using several software packages for Python created by the author (collectively called *Modular*), which remain freely available online (http://github.com/ctogle/modular) and are discussed at length in Chapter 5.

*Modular* depends heavily on several community supported Python packages which are worth noting. Firstly, Cython [6] provides a robust mechanism to translate Python source code into C code, compiling it into an extension that is then available to Python. By providing additional information in the Python source file, the resulting library can be greatly optimized, providing code that runs at nearly C speeds but written almost as easily as Python. Performing this extra optimization on bottlenecks of the simulation code allows Python to effectively compete with compiled languages such as C++, which is of great utility to the scientific community. NumPy [22] is a Python package which provides additional data structures that operate very efficiently and provides extensive functionality via an implementation of an array. Written partially in Cython itself, NumPy is a de facto expectation in scientific computing with Python. Packages such as SciPy [54] leverage the data structures of NumPy, providing robust and efficient implementations of routines to measure many statistical quantities. Lastly the Python package MPI4Py [19] provides an implementation of the message-passing-interface (MPI) commonly used in computation on clusters of computers. This allows for incredible speedups that can make otherwise impossible computational problems rather easy. Generation of the data in later chapters often required the use of MPI4Py to leverage computing clusters, and MPI4Py allows *Modular* to facilitate their use for simulation and analysis.

### 2.3 Queueing Theory

Queueing theory is a formalism concerned with representing systems of servers and customers, where servers perform discrete tasks for customers according to a set of rules. These rules can govern anything from the number of servers or customers in the system to whether or not a task can be interrupted to complete a higher priority task. Queueing theory characterizes the buildup of queued tasks and how networks of servers can experience bottlenecks in processing affecting the statistics associated with task completion. It is naturally discretized, making it appropriate for low copy-number systems where noise can strongly influence dynamics and resulting statistics. Having existed for more than a century, queueing theory is a mature formalism, with an origin in describing telephonic traffic, which is both discrete and stochastic by nature [18, 44, 50, 71].
2.3.1 Queueing Regimes

Queueing theory frequently leverages three regimes to describe the state of traffic on a network of servers. The “underloaded” regime is characterized by more than sufficient processing capacity of servers to prevent bottlenecks from forming, which prevents queues of tasks from forming. The “overloaded” regime is denoted by long queues of incomplete tasks and constant processing of tasks by servers. Lastly the “balanced” regime is characterized as the transition between the other two regimes. Here there is barely enough processing capacity of the servers to prevent the formation of long queues [18, 44, 50, 71]. Queueing theory often shows that statistics associated with server networks can change drastically as systems transition between regimes. It is sometimes the goal to identify conditions which describe this transition by measuring the point of balance with respect to some variable quantity [18, 50, 71].

As a demonstration of such a measurement, consider a system consisting of a single job class $X$ which arrives at a server according to a Poisson process. Jobs will thus enter the system at exponentially distributed points in time. Jobs are processed independently by the servers at a rate $\mu$. This situation is representable using the following chemical reaction equations suitable for simulation using the Gillespie algorithm.

\[
\begin{align*}
\emptyset & \xrightarrow{\lambda} X \quad \text{with propensity} \quad \lambda \\
X & \xrightarrow{\mu} \emptyset \quad \text{with propensity} \quad \mu X \\
\mu &= \frac{\omega}{\epsilon + X}
\end{align*}
\]

The parameter $\epsilon$ is related to the possibility that a server is unoccupied in the presence of incomplete jobs and is small corresponding to efficient servers. As previously suggested, a variation of the job arrival parameter $\lambda$ may transition the system across balance, resulting in a significant change in dynamics. In particular, $\lambda < \omega$ results in nearly zero incomplete tasks from ever accumulating. $\lambda > \omega$ results in the unbounded accumulation of jobs. These situations are the underloaded and overloaded cases respectively, with $\lambda = \omega$ identifying the point of balance (see Figure 2.4), near which fluctuations of the count of incomplete jobs are relatively large [18, 71].

2.3.2 Applicability to Biology

More recently, queueing theory has proven to be appropriate for systems considered by cell biology [18, 50, 53, 71]. In particular molecules exist in integer amounts within cells, and molecular counts of proteins are often sufficiently low to observe differences between discrete and continuous descriptions. As much as 75% of proteins found in $E. \ coli$ have a
Figure 2.4: **Demonstration of queueing regimes.** A measurement of the mean job count after some period of time as a function of the production parameter $\lambda$ demonstrates the three regimes of queueing theory. Measurement of the job count $X$ was made at 100 time units using ensembles of 10,000 realizations. $\omega$ and $\epsilon$ were fixed at 10 and 0.00001 respectively.

copy number below 250. The copy number of chromosomes is as low as 1 or 2, while mRNA molecules are often found in the tens. Many systems considered by cell biology possess some component which is typically of low copy number, possibly requiring discretization to faithfully represent [11, 38, 40, 104]. Small molecular fluctuations at the DNA and mRNA level propagate to protein levels, amplifying the influence of noise in such systems.

Queueing theory will be applied to biological networks in the following chapters, often considering a protease to be a server and cellular degradation targets to be tasks. It will be demonstrated that the transition across the balance point is marked by profound changes in steady state cellular concentrations. In some systems, these changes are essential to the function of particular gene networks. It will be shown that activity with proteases can couple subsystems of cells, resulting in statistics of molecular counts which depend strongly on the queueing regime the system occupies.

As a simple example of this, consider the following network of chemical reactions which represent constant zeroth-order production of two proteins, cell division resulting in dilution, and first-order degradation by an enzyme. For analogy with customers and servers, consider each protein class to be a distinct type of task; the molecular count of each protein is the number of such tasks which have been issued to the server. The server is the enzyme which completes each task using one reaction, and dilution represents the situation where a customer decides to go elsewhere rather than wait for the server.
This network can be simulated using an implementation of the Gillespie algorithm. This is an effective representation for proteolytic degradation [71, 80], compact and efficient for stochastic simulation, and will be adopted in varying forms throughout the following chapters. Here, $\gamma$ must be much smaller than $\omega$ as degradation occurs far more frequently than cell division. $\epsilon$ is related to the likelihood that the protease will be unoccupied in the presence of a target protein and is consequently small. As previously suggested, a variation of the production parameters $\lambda_1$ or $\lambda_1$ transitions the system across balance, resulting in a significant change in dynamics. The above system has been chosen because it is sufficient to demonstrate a statistical phenomenon which will be particularly relevant throughout the following chapters.

By simulating a sufficiently long amount of time, steady state statistics of this system can be measured. In the overloaded regime the accumulation of proteins would prevent the system from ever reaching a steady state, if not for the first-order dilution reaction which effectively sets the maximum protein count that can accumulate. Simulating the system for a range of parameter values while measuring the steady state Pearson correlation coefficient between the molecular counts of the two proteins exhibits a maximum near the balance point which is known as correlation resonance [18, 71]. This situation is of interest as it indicates that the two protein concentrations are coupled, and that this coupling is strongest as the system transitions between the underloaded and overloaded regimes (see Figure 2.5). This transition is also marked by sharp changes in the steady state mean counts of each protein [18, 71].

It is necessary to ask if this situation actually occurs within natural systems. Within many bacteria, there are specialized transcription factors called sigma factors associated with cellular responses to various environmental stressors such as large variations in temperature or starvation [34]. Under healthy conditions they are typically degraded by proteases without significant opportunity to increase the expression of stress response genes. During stress, proteolytic machinery may experience an increase in workload as the cell produces erroneous proteins which must be degraded. This increase in workload forces the system into the overloaded regime where sigma factors are not as efficiently degraded, resulting in expression of the genes associated with rescuing the cell [18, 34]. Competition for degradation, in particular the formation of queues associated with the balanced regime, can thus serve as a
Figure 2.5: **Demonstration of correlation resonance.** Three distinct measurements of steady state correlation resonance taken using ensembles of 20,000 realizations are shown for a simple network with two protein classes competing for one protease class. $\lambda_1$ is varied between 1 and 50 with $\lambda_2$ fixed at 10 (green), 25 (blue), and 40 (red). $\omega$, $\epsilon$, and $\gamma$ were fixed at 50, 0.00001, and 0.01 respectively. For each curve, correlation resonance is identified with dashed black lines where $\lambda_1 + \lambda_2 = \omega$.

triggering mechanism by selectively permitting accumulation of sigma factors during stress [18].
Chapter 3

Proteolytic Crosstalk in Multi-Protease Networks

This chapter presents published work with minor formatting modifications [78].

3.1 Abstract

Processive proteases, such as ClpXP in \textit{E. coli}, are conserved enzyme assemblies that can recognize and rapidly degrade proteins. These proteases are used for a number of purposes, including degrading mistranslated proteins and controlling cellular stress response. However, proteolytic machinery within the cell is limited in capacity and can lead to a bottleneck in protein degradation, whereby many proteins compete (“queue”) for proteolytic resources. Previous work has demonstrated that such queueing can lead to pronounced statistical relationships between different protein counts when proteins compete for a single common protease. However, real cells contain many different proteases, e.g. ClpXP, ClpAP, and Lon in \textit{E. coli}, and it is not clear how competition between proteins for multiple classes of protease would influence the dynamics of cellular networks. In the present work, we theoretically demonstrate that a multi-protease proteolytic bottleneck can substantially couple the dynamics for both simple and complex (oscillatory) networks, even between substrates with substantially different affinities for protease. For these networks, queueing often leads to strong positive correlations between protein counts, and these correlations are strongest near the queueing theoretic point of balance. Furthermore, we find that the qualitative behavior of these networks depends on the relative size of the absolute affinity of substrate to protease compared to the cross affinity of substrate to protease, leading in certain regimes to priority queue statistics.
3.2 Introduction

Queueing theory is a mathematical formalism originally used to describe telecommunications networks, where finite processing resources naturally led to bottlenecks and potentially long waiting lines (“queues”) [44]. A host of quantitative and qualitative tools have since emerged to analyze the traffic for a variety of queueing networks, addressing wide classes of discrete stochastic models for bottlenecks. Interestingly, queueing theory promises to also be a natural language for a variety of biochemical networks, where enzymatic and material resources are often limited, and where low copy number effects and natural noise play important roles in chemical processing [7, 26, 27, 81, 91, 104]. Queueing theory has recently been used to characterize several biological systems, in particular systems involving enzymic chemical kinetics in metabolism or proteolytic pathways [18, 50, 53, 56, 63, 71]. A major strength of queueing theory is that it intuitively links several regimes in queueing theory (overloaded, underloaded, and balanced regimes) to qualitatively different system dynamics [18, 36, 71]. In particular, crossing the balanced regime has been shown to exhibit a robust phenomenon known as correlation resonance, where the two substrates exhibit a surprisingly high Pearson correlation coefficient [18, 71, 72].

Queueing can be a useful design principle in biochemical networks, because it permits indirect control of one substrate through direct control of the other [36]. A natural example occurs during cellular stress response, where it has been shown in *E. coli* that the buildup of stress response sigma factors follows increased translational stress by transitioning the cell between the underloaded and overloaded regimes [18, 34, 74]. This overload occurs as stress in the environment such as nutritional starvation causes misfolded or partially constructed proteins to accumulate. These faulty proteins are then targeted to enzymes for degradation using some recognition scheme. In *E. coli*, ssrA tagged substrates as well as the stationary phase sigma factor $\sigma^s$ are targeted to the proteases ClpXP and ClpAP [10, 16, 18, 33, 43, 48, 73, 86, 107], resulting in the overload of these degradation pathways which results in an increase in the level of $\sigma^s$. Thus the cell uses a bottleneck in degradation processing to trigger a stress response within the cell that aids in coping with environmental stress. A variety of synthetic biological systems also include queueing prominently in their design. For example, the mechanism behind a variety of synthetic gene oscillators is linked to delayed negative feedback in conjunction with queueing in degradation (saturated degradation) [8, 80]. Coupling of substrates via degradation queueing has also been used in a synthetic circuit to synchronize genetic clocks, providing rapid control more than an order of magnitude faster than transcriptional coupling mechanisms [85].

Most, if not all, studies of proteolytic queueing have focused on the competition of different protein substrates for a common protease, ClpXP. However, *E. coli* contains many ATP-dependent proteases, such as Lon, FtsH, ClpYQ, ClpXP, and ClpAP [16, 33, 43, 51, 61, 109], with Lon, ClpXP, and ClpAP together accounting for approximately 70-80% of ATP-dependent degradation in the cell [61]. It is reasonable to suspect that a variety of proteins could be targeted to more than one protease (with differing affinities) in the natural context.
Modulation of protease affinity is owed to several layers of control within the cell. For example, special adaptor proteins that aid in the delivery of specific substrates may have spurious affinities to multiple proteases. This suggests that indirect coupling of seemingly independent subsystems via proteolytic activity may be a common phenomenon, possibly even useful to a cell where a correlated response might present an advantage such as promoting coherent signals. Such multi-protease crosstalk presents a particular problem for synthetic systems, where poor insulation of synthetic circuits can be a major impediment to scalability of synthetic circuits in a cell [21, 89].

In this work, we theoretically investigate the phenomenon of multi-protease queueing using analytical and numerical approaches. Previous work involving proteolytic crosstalk focused on single pathway dynamics [18, 71], where two substrates were coupled via competition for a common protease. Here we focus primarily on dynamics involving two or more classes of substrate (e.g., protein) simultaneously processed by two distinct enzymatic (e.g., proteolytic) pathways. We show that even when different substrates exhibit preference for their own respective enzymes, strong coupling can occur due to weakly shared enzymatic activity. Furthermore, statistical dependence between the abundance of substrate species is closely linked to multi-enzyme queueing theoretic regimes, and in particular, the phenomenon of correlation resonance is observed near the queueing theoretic balance point. Our work as a whole further establishes queueing theory as an intuitive yet quantitative framework for enzymatic processing networks.

An outline of this work is as follows. Methods are explained in Section 3.3, which in particular includes in Section 3.3.1 a discussion of the core features of the model used to explore multi-protease queueing. Results appear in Section 3.4, with results discussing general queueing regimes appearing in Section 3.4.1, results based on the extensive simulation of several related multi-protease models in Section 3.4.2, and results on queueing-based synchronization between gene oscillators in Section 3.4.3. Conclusions appear in Section 3.5.

### 3.3 Methods

#### 3.3.1 Multi-Enzyme Crosstalk Model

We explore reaction network models where multiple classes of substrate are processed by multiple classes of enzyme. Our discussion applies readily to proteins that are degraded by proteases, though we will often keep the language of our discussion more general. These models can be graphically represented using bipartite graphs between substrate and enzyme classes. We label these networks \((M, N)\) when \(M\) classes of substrate are processed by \(N\) classes of enzyme. Figure 3.1 provides representative graphs for the multi-enzyme reaction networks treated in this work.

Let \(X_j\), where \(1 \leq j \leq M\), represent the counts of \(M\) distinct classes of substrate. Assuming
Figure 3.1: Schematic of simple crosstalk networks. A schematic of the queueing networks we consider in this work, labeled by \((M, N)\) to indicate \(M\) substrate classes and \(N\) enzyme classes. The \((1, 1)\) system represents a standard single class queueing model used in introductory queueing theory, while the remaining systems exhibit richer dynamics. In our model, the dynamical variables are the counts of substrate, and dimension of the dynamics is thus set by \(M\). We do not explore the \((1, 2)\), \((1, 3)\), or \((2, 3)\) systems, since in our model, these do not appear to generate qualitatively new results beyond the depicted systems.

There are \(N\) distinct classes of enzyme, let \(\omega_{ij}\), where \(1 \leq i \leq N\), represent the maximum processing rate of substrate class \(j\) by enzyme class \(i\). Each distinct class of enzyme possesses an affinity for each class of substrate. Define \(\kappa_{ij}\) as the affinity of substrate class \(j\) for enzyme class \(i\). Each enzyme in this model also has a relatively small constant \(\epsilon_i\) that scales the overall affinity of substrate to enzyme. Combining reactions due to production, enzymatic processing, and dilution (due to cell growth and division) of each class of substrate yields a reaction network similar to those explored previously [71]

\[
\begin{align*}
\emptyset & \xrightarrow{\lambda_j} X_j \\
X_j & \xrightarrow{\gamma} \emptyset \\
X_j & \xrightarrow{\mu_{ij}} \emptyset
\end{align*}
\]

(3.1)

\(\text{(3.2)}\)

\(\text{(3.3)}\)

where each \(\mu_{ij}\) is a rate constant defined by the function

\[
\mu_{ij} = \frac{\omega_{ij} \kappa_{ij}}{\epsilon_i + \sum_{k=1}^{M} \kappa_{ik} X_k}
\]

(3.4)
with associated enzymatic velocity (propensity)

\[ v_{ij} = \mu_{ij} X_j \]  (3.5)

Here, \( \lambda_j \) is a constant production rate for substrate class \( j \), and the dilution rate \( \gamma \) is shared by every substrate class. With the definition of \( \mu_{ij} \) in Eq. 3.4, the reaction rate in Eq. 3.3 corresponds to \( N \) enzymatic processing reactions for substrate class \( j \). When each possible value of \( i \) and \( j \) is considered, reactions in Eqs. 3.1-3.4 represent \( M (N + 2) \) independent birth-death reactions. For simulation, we only further consider systems where \( N \leq 2 \) and \( M \leq 3 \). We assume that all reactions occur with exponentially-distributed reaction times. We do not explicitly model enzyme state, though we do not believe our qualitative results should sensitively depend on this assumption [18]. To simplify our discussion, we use the same symbols (e.g. \( X_1 \)) for species counts and species labels.

Preliminary analysis of the reaction network in Eqs. 3.1-3.4 is assisted by a deterministic \( M \)-dimensional ODE approximation

\[ \frac{dX_j}{dt} = \lambda_j - X_j \left( \gamma + \sum_{i=1}^{N} \mu_{ij} \right) \]  (3.6)

with \( M \) associated steady state equations:

\[ \lambda_j = X_j \left( \gamma + \sum_{i=1}^{N} \mu_{ij} \right) \]  (3.7)

Equations 3.6-3.7 often provide a good qualitative description of the mean substrate count dynamics and steady state, especially when mean substrate count is high. In this work, we will apply the ODE approximation only to facilitate intuitive understanding of qualitative dynamics in queueing systems.

### 3.3.2 Degradate-and-Fire Oscillator Model

Our work in Section 3.4.3 considers how multi-enzyme queueing can couple and thus synchronize models for gene oscillators. The basic model is a variant of a degrade-and-fire oscillator, which has frequently been used as a model for certain synthetic gene oscillators that depend on rapid proteolytic degradation and delayed negative feedback [69,70,80,100]. The model includes maturation reactions in the production of a regulatory protein \( X_j \), the effect of which is a delay in production. We use ten such maturation reactions in our model to generate a delay in production with a sufficiently sharp distribution, which is required for oscillations to occur [92]. The maturation rate \( \beta \) is chosen to be the same for each maturation reaction. In addition to assuming a delay in the production of the mature protein \( X_j \), we assume that \( X_j \) represses the production of its nascent form. These two features (delayed
production and auto-repression) constitute a delayed negative feedback, which is sufficient for stable oscillations of the protein count $X_j$ \cite{87,100}.

Our model for an oscillator with index $j$ is defined by the set of reactions in our simple queueing model (see Section 3.3.1 above), but we replace the production reaction for $X_j$ with the reactions

\begin{align}
\emptyset & \xrightarrow{\lambda_j} r_{j,1} \\
r_{j,1} & \xrightarrow{\beta} r_{j,2} \\
\vdots & \notag \\
r_{j,10} & \xrightarrow{\beta} X_j
\end{align}

where $\lambda_j$ is now a state-dependent production rate for $r_{j,1}$ that includes repression by $X_j$

$$\lambda_j = \frac{\alpha_j}{1.0 + \left( \frac{X_j}{C_0} \right)^2}$$ \hspace{1cm} (3.12)

Using this reaction scheme, any particular substrate variable $X_j$ in the simple queueing model in Section 3.3.1 can be substituted by an oscillator model.

For a single oscillator with a single protease, this model generates alternate firing (rapid production of protein) and degradation (proteolytic clearance of protein) phases, hence the name degrade-and-fire \cite{69}. The period of oscillation is largely determined by the time to degrade the protein produced during the firing phase. Synchronous oscillations within or across cells can be generated by either controlling the amplitude of firing \cite{9,70} or by modulating the duration of degradation \cite{85}, and it is the latter that we study in the context of queueing-based synchronization.

### 3.3.3 Simulation Algorithm

All stochastic simulations were performed using a custom implementation of the Gillespie algorithm, run either on a personal computer or a computing cluster. These simulations were executed and analyzed using two of our Python packages, with recent versions available at \url{www.github.com/ctogle/modular} and \url{www.github.com/ctogle/gillespiem}.

### 3.3.4 Statistical Analysis

Unless otherwise specified, all steady state measurements were obtained for ensembles of at least 20,000. Models were simulated for a time duration of 499 for simple queue models and 999 for oscillator models to sufficiently eliminate transient behavior. Steady state measurements were taken over the last 20% of the time simulated in every case, during which time transients in oscillator statistics appeared to be minimal.
Correlation was computed using the Pearson correlation coefficient, which is a standard metric for statistical dependency between two variables [7, 18, 26, 36, 71].

### 3.3.5 Non-equilibrium Steady State Analysis of (2, 2) Queueing Network Behavior

For a preliminary investigation of the non-equilibrium properties of multi-enzyme networks, we applied the thermodynamic formalism detailed by Schnakenberg [96], which allows for a description of network dynamics based on thermodynamic cycles. To perform this analysis, we solved for the steady state of the master equation describing the system, where each state in the master equation is defined by a particular pair of counts \((X_1, X_2)\) [41]. We numerically found the steady state of the master equation’s evolution operator using custom code that leverages sparse linear algebra routines in SciPy. The evolution operator was approximated using a finite system defined by the two-dimensional grid \(0 \leq X_1 \leq 100\) by \(0 \leq X_2 \leq 100\).

Define the steady state current \(J_1(X_1, X_2)\) between points \((X_1, X_2)\) and \((X_1 + 1, X_2)\), and define the steady state current \(J_2(X_1, X_2)\) between points \((X_1, X_2)\) and \((X_1, X_2 + 1)\). This is defined in the usual way [96]. To perform the cycle analysis, a set of fundamental cycles must be determined. For this, we associate a thermodynamic cycle \(C(X_1, X_2)\) to each system state labeled by counts \((X_1, X_2)\)

\[
C(X_1, X_2) = (X_1, X_2) \rightarrow (X_1 + 1, X_2) \rightarrow (X_1 + 1, X_2 + 1) \rightarrow (X_1, X_2 + 1) \rightarrow (X_1, X_2)
\]

(3.13)

The steady state cycle current \(J_{\text{cycle}}(X_1, X_2)\) can be solved iteratively based on steady state currents

\[
J_{\text{cycle}}(X_1, 0) = J_1(X_1, 0) \quad (3.14)
\]

\[
J_{\text{cycle}}(X_1, X_2) = J_1(X_1, X_2) + J_{\text{cycle}}(X_1, X_2 - 1) \quad (X_2 > 0) \quad (3.15)
\]

The cycle potential \(-\Delta G(X_1, X_2)\) is defined as the negative cycle free energy [96], which only depends on the sum of log-ratios of reaction rates, and which does not depend on a knowledge of steady state probabilities or currents. A great strength of the cycle potential is that if \(\Delta G(X_1, X_2) = 0\) for all cycles in the system, which can be determined without solving for steady state, then the system is reversible at steady state and has a simple (detailed balance) steady state solution. The cycle power (or cycle entropy production) is defined as \(P = -\frac{1}{2} J_{\text{cycle}} \Delta G_{\text{cycle}}\). This power is zero for all cycles in a reversible system, and a summation over all cycle powers leads to the total entropy production of the network [96].
3.4 Results and Discussion

3.4.1 Queueing Regimes for Multi-Enzyme Models

One of the great strengths of queueing theory is that the qualitative behavior of a queueing network is largely determined by the queueing regime it occupies, and determination of the queueing regime often depends only on testing simple inequalities involving parameters. The major regimes are **underloaded**, **overloaded**, and **balanced**, and each regime produces its own characteristic statistics. For our models, the system is underloaded when the rate of processing by enzyme is always sufficient to rapidly clear newly produced substrate, thus keeping the mean count of substrate small. In contrast, the system is overloaded when enzymes cannot process substrates quickly enough, and the mean count of substrate continues to increase to large values. Indeed, the mean count would become infinite at infinite time in an overloaded system if other reactions, e.g. dilution, did not eventually become important. Finally, the boundary case between underloaded and overloaded is termed balance, where very strong fluctuations in count occur characteristic of a continuous phase transition. For example, a queueing network crossing balance has been shown to exhibit a continuous phase transition with a correspondingly strong positive “correlation resonance” between substrate counts [18, 71]. For the sake of demonstration, assume a (1, 1) system with a substrate arrival rate $\lambda$ and enzymatic velocity $\omega$.

For such a system, the underloaded, overloaded, and balanced regimes would correspond to $\lambda < \omega$, $\lambda > \omega$, and $\lambda = \omega$, respectively.

The dynamical behavior of the general model defined by Eqs. 3.1-3.4, or indeed even determination of its queueing regime, can be surprisingly complex. For example, interesting results can arise for a (2, 2) system where

\begin{align}
\omega_{11} &= \omega_{22} = \omega_1 \\
\omega_{21} &= \omega_{12} = \omega_2 \\
\kappa_{11} &= \kappa_{22} = \kappa_1 \\
\kappa_{21} &= \kappa_{12} = \kappa_2 \\
\epsilon_1 &= \epsilon_2 = \epsilon
\end{align}

with $\omega_1 \gg \omega_2$ but $\kappa_2 \gg \kappa_1 \gg \epsilon$. That is, the preferred (in terms of affinity) enzyme for a given substrate is actually the slowest enzyme for that substrate. It can then be shown that the qualitative dynamics of the network when either $\lambda_1 = 0$ or $\lambda_2 = 0$ are quite different than the case when $\lambda_1 > 0$ and $\lambda_2 > 0$ together, as the latter case has a drastically reduced total enzymatic velocity (approximately $2\omega_2$) relative to the total enzymatic velocity in the former case (approximately $2\omega_1$). A consequence is that the queueing regime depends sensitively on whether or not both classes of substrate are expressed simultaneously. While analogous cases do indeed occur in biology, e.g. as with a classical competitive inhibitor, it is arguable whether a queueing formalism is the most useful approach in such cases.

We will now restrict our attention to systems that are more regular with respect to their
queueing regimes, and for which we believe the queueing approach is more transparent than alternative approaches. There exist at least two representative special cases for our model that facilitate qualitative analysis based on a queueing approach. These two major classes of systems are characterized by symmetries in the matrix $\omega_{ij}$ and allow for the straightforward identification of the major queueing regimes based on inequalities relating production rates $\lambda_j$ and maximum processing rates $\omega_{ij}$ of substrate. The prediction emerging from this analysis is that major changes in the state of the queueing network, here determined by the counts of substrate, should coincide with changes in the queueing regime. We will demonstrate the influence of these queueing regimes numerically using particular models in Section 3.4.2.

One possible assumption to simplify our model is that the maximum processing rate for enzymes only depends on the class of substrate but not the class of enzyme, i.e. $\omega_{ij} = \omega_j$. This allows for the use of a workload formalism, which greatly simplifies the analysis of many queueing theories [47, 105]. The workload $W$ is intuitively the mean time to enzymatically process all substrate presently in the system, assuming substrate is rapidly associated with enzyme. A straightforward definition for $W$ can then be written as

$$W = \sum_{j=1}^{M} \frac{X_j}{\omega_j}$$  \hspace{1cm} (3.21)

where $1/\omega_j$ is the mean time for an enzyme to process substrate class $j$ once bound to enzyme. The workload $W$ in Eq. 3.21 is useful for a number of analytic purposes, but the simplest is that the workload increases without bound only when one or more of the substrate counts increases without bound. This leads to an easy way to identify queueing regimes. First ignoring dilution to lowest order by setting $\gamma = 0$, the deterministic approximation in Eq. 3.6 would then predict

$$\frac{dW}{dt} = \sum_{j=1}^{M} \frac{1}{\omega_j} \frac{dX_j}{dt}$$

$$= \sum_{j=1}^{M} \left( \frac{\lambda_j}{\omega_j} - \frac{\lambda_j}{\omega_j} X_j \frac{\sum_{i=1}^{N} \mu_{ij}}{\sum_{j=1}^{M} \kappa_{ij} X_j} \right)$$

$$= \sum_{j=1}^{M} \frac{\lambda_j}{\omega_j} - \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{M} \kappa_{ij} X_j}{\sum_{j=1}^{M} \kappa_{ij} X_j}}{\epsilon_i + \frac{1}{\sum_{j=1}^{M} \kappa_{ij} X_j}}$$

$$= \sum_{j=1}^{M} \frac{\lambda_j}{\omega_j} - \frac{\sum_{i=1}^{N} \frac{1}{\epsilon_i / \sum_{j=1}^{M} \kappa_{ij} X_j}}{1} + 1$$  \hspace{1cm} (3.25)

where we assume $\sum_{j=1}^{M} \kappa_{ij} X_j > 0$ for all permissible values of $i$. For sufficiently large counts $X_j$, this equation is approximately

$$\frac{dW}{dt} \approx \sum_{j=1}^{M} \frac{\lambda_j}{\omega_j} - N$$  \hspace{1cm} (3.26)
where the first term is the rate that workload is added to the system, while the second term is the rate that workload is removed from the system. The underloaded regime corresponds to \( \sum_{j=1}^{M} (\lambda_j/\omega_j) < N \), since workload systematically decreases when substrate counts are large. Analogously, the overloaded regime corresponds to \( \sum_{j=1}^{M} (\lambda_j/\omega_j) > N \), while balance corresponds to \( \sum_{j=1}^{M} (\lambda_j/\omega_j) = N \). As a result, the transition between different queueing regimes is more sensitive to substrate that takes longer to process (smaller associated \( \omega_j \)). A proteolytic system degrading tagged titin, for example, may be strongly influenced by titin’s low degradation velocity [4].

A slightly different and perhaps more reasonable simplification is to instead assume that the maximum processing rate for enzymes is dependent only on the enzyme class rather than the substrate class, i.e. \( \omega_{ij} = \omega_i \). Such a first approximation may be reasonable in natural proteolytic networks, which have been shown to exhibit a range of enzyme-dependent rates [46]. This allows the network to be analyzed in terms of overall production rate and degradation rate of substrate, leading in particular to an understanding of queueing regimes.

The total production reaction velocity \( \Lambda \) (across all substrate classes) is

\[
\Lambda \equiv \sum_{j=1}^{M} \lambda_j
\]

while the total enzyme reaction velocity \( v_{\text{tot}} \) (across all substrate classes) is

\[
v_{\text{tot}} \equiv \sum_{j=1}^{M} \sum_{i=1}^{N} \mu_{ij} X_j
\]

\[
= \sum_{i=1}^{N} \omega_i \cdot \frac{\sum_{j=1}^{M} \kappa_{ij} X_j}{\epsilon_i + \sum_{k=1}^{M} \kappa_{ik} X_k}
\]

\[
= \sum_{i=1}^{N} \omega_i \cdot \frac{1}{\left(\frac{\epsilon_i}{\sum_{j=1}^{M} \kappa_{ij} X_j}\right) + 1}
\]

where we assume \( \sum_{j=1}^{M} \kappa_{ij} X_j > 0 \) for all enzyme indices \( i \). It is then straightforward to see that the sum \( \Omega \) of enzymatic velocities

\[
\Omega \equiv \sum_{i=1}^{N} \omega_i
\]

sets the bound

\[
v_{\text{tot}} \leq \Omega
\]

for any system state, and this bound is approached from below for large substrate counts. Qualitative regimes for the reaction network in Eqs. 3.1-3.4 can then be found by comparing
the magnitude of $\Lambda$ to $\Omega$. Ignoring dilution to lowest order, the case when $\Lambda < \Omega$ can be identified as the underloaded regime, because on average the system processes substrate faster than substrate arrives. Analogously, the overloaded regime is identified with $\Lambda > \Omega$, and balance is identified with $\Lambda = \Omega$. As before with the workload formalism, the queueing regime depends on the relative magnitude of production rates to processing rates.

### 3.4.2 Numerical Analysis of Queueing-Based Crosstalk for Simple Queueing Networks

The discussion in Section 3.4.1 demonstrated that the form of the matrix $\omega_{ij}$ has a major impact on the boundaries of queueing regimes that determine the qualitative behavior of queueing systems. However, we are also interested in to what degree the values of $\kappa_{ij}$ can influence system dynamics within the different queueing regimes. This quantitative information is essential, for example, when discussing the degree of insulation between different gene networks that partially share proteolytic pathways, such as networks in *E. coli* that leverage both ClpXP and ClpAP proteases.

To this end, the following subsections explore special cases for the $(2, 1)$, $(2, 2)$, and $(3, 2)$ network structures. These three systems demonstrate that a queueing phenomenon, correlation resonance [18, 71], also occurs in the multi-enzyme setting and is associated with the balanced regime. Furthermore, these systems illustrate that strong statistical relationships between substrate counts can persist even for weak multi-enzyme crosstalk, i.e. when certain substrate-enzyme affinities are small.

For the sake of brevity for the remainder of the manuscript, we make the assumption that $\omega_{ij} = \omega$, i.e. that all maximum degradation velocities are equal in value (relaxing this assumption does not appear to drastically impact our results, as we will discuss.) The conditions that $\Lambda < N\omega$, $\Lambda > N\omega$, and $\Lambda = N\omega$ then correspond to the underloaded, overloaded, and balanced regimes, respectively, where we define $\Lambda$ by Eq. 3.27. We also fix each $\epsilon_i$ to a single value $\epsilon$.

Our computational investigation requires that we eventually commit to numerical values for the parameters characterizing the enzyme network. In the case of proteases, a few of these parameters have been estimated either *in vitro* or *in vivo*, such as for the highly important Lon protease [45]. However, these parameters - especially affinities - can be strongly dependent on the chemical composition. One example is for ssrA-tagged proteins, which have a relative affinity (and absolute affinity) to ClpXP that is strongly affected by the adaptor protein SspB [32]. Thus, rather than simulating models for parameter values derived from the literature, we choose to explore scans through parameter space that we consider illustrative. Parameters are given in arbitrary units to emphasize our choice to avoid particular experimental systems. Dilution is set to be relatively slow, characterized by the small value $\gamma = 0.0231$ that corresponds to a half-life of 30 time units. The degradation
velocity \( \omega \) is chosen to be orders of magnitude larger than \( \gamma \), such that dilution is weak relative to enzymatic degradation at low substrate copy number. Substrate production rates are scanned over a range set by the magnitude of \( \omega \). The relative affinity of substrate to enzyme is allowed to vary over a wide range, since this is the primary variable we wish to explore. The absolute affinity of substrate to enzyme is typically assumed to be strong \((\epsilon = 10^{-5})\), but we also explore results for moderate affinity \((\epsilon = 1 \text{ or } \epsilon = 10)\).

**Coupled Substrates of a \((2, 1)\) System**

The \((2, 1)\) case was partially explored previously in application to multi-substrate queueing in the case of a single protease ClpXP \([18, 71]\), and so we will not explore this case in the same detail as for \((2, 2)\) and \((3, 2)\) systems appearing in the following subsections. We assign values to the affinities using a parameter \(\eta\)

\[
\begin{align*}
\kappa_{11} &= \eta \\
\kappa_{12} &= 1 - \eta
\end{align*}
\]

where \(0 \leq \eta \leq 1\). Equal affinity for the substrates occurs when \(\eta = 0.5\), in which case we recover the result that trajectories for substrate count can exhibit extremely correlated behavior near the balance point because they share a common bottleneck (see Fig. 3.2A) \([71]\). Similar behavior is observed for \(\eta = 0.2\) (see Fig. 3.2B). More generally in the vicinity of the balance point, a peak in the \(X_1, X_2\) correlation exists. This is observed by varying the production rate of one substrate such that the balance point is crossed (see Fig. 3.2C). Of note is that the parametric position of this correlation resonance in relation to balance is relatively insensitive to the substrate affinity \(\eta\), while the magnitude of the correlation resonance depends on \(\eta\) weakly. Thus, we predict that even if the affinity of one substrate for the enzyme is relatively weak compared to another substrate (e.g. \(\eta = 0.2\)), substantial statistical dependence between the two substrate counts may occur near balance.
Figure 3.2: **Results for a (2, 1) queueing network.** Results for a (2, 1) queueing network (see Section 3.4.2 for details). (A) For this queueing network at balance, with equal sharing of protease ($\eta = 0.5$), the counts of two substrate molecules ($X_1$ and $X_2$) exhibit strongly correlated dynamics due to competition for enzyme processing. (B) Strongly correlated trajectories persist for unequal substrate affinities ($\eta = 0.2$). (C) The correlation coefficient between $X_1$ and $X_2$ as a function of $\eta$ and $\lambda_2$ shows that a strong correlation resonance appears very near the queueing theoretic balance point ($\lambda_1 + \lambda_2 = \omega$, dashed line) for a wide range of $\eta$ values. This result extends the investigation of correlation resonance observed for the case $\eta = 0.5$ [18, 71]. Parameters are $\lambda_1 = 15$, $\gamma = 0.0231$, $\omega = 30$, and $\epsilon = 10^{-5}$, with $\lambda_2$ and $\eta$ variable.
Coupled Substrates of a \((2, 2)\) System

The \((2, 2)\) multi-enzyme system is the first system where two enzymatic networks can exist in parallel and in principle be independent of one another. This case is of particular interest as a model for insulation between different enzymatic pathways, e.g. between ClpXP and ClpAP degradation pathways. We consider a particular \((2, 2)\) system, where the cross-affinities of substrates for enzymes are expressed in terms of a coupling parameter \(\eta\)

\[
\begin{align*}
\kappa_{11} &= 1 - \eta \\
\kappa_{21} &= \eta \\
\kappa_{12} &= \eta \\
\kappa_{22} &= 1 - \eta
\end{align*}
\]

where \(0 \leq \eta \leq 1\). This parameterization leads to zero crosstalk in the cases where \(\eta = 0\) or \(\eta = 1\). When \(\eta = 0.5\), the inter-substrate coupling is maximized, as each substrate has no preference for a particular degradation pathway and readily shares pathways with the other.

Expressing the enzymatic reactions of each substrate as a single reaction whose rate is the sum of the rates of each degradation reaction for that substrate yields the following relatively simple reaction network

\[
\begin{align*}
\emptyset & \xrightarrow{\lambda_j} X_j \\
X_j & \xrightarrow{\gamma} \emptyset \\
X_j & \xrightarrow{\mu_j} \emptyset
\end{align*}
\]

for \(j \in \{1, 2\}\), and using the definitions

\[
\begin{align*}
\mu_1 &= \frac{\omega (1 - \eta)}{\epsilon + (1 - \eta) X_1 + \eta X_2} + \frac{\omega \eta}{\epsilon + \eta X_1 + (1 - \eta) X_2} \\
\mu_2 &= \frac{\omega \eta}{\epsilon + (1 - \eta) X_1 + \eta X_2} + \frac{\omega (1 - \eta)}{\epsilon + \eta X_1 + (1 - \eta) X_2}
\end{align*}
\]

Because of the symmetry in the above representation, investigating \(\eta\) beyond 0.5 yields no new information about the system, as it effectively exchanges the identities of the substrates, so we only consider the parameter space of \(\eta\) bounded such that \(0 \leq \eta \leq 0.5\).

Since we choose \(\omega_{ij} = \omega\) for all \(i, j\), varying the parameter \(\eta\) does not change the overall processing rate of any class of substrate; it only changes the manner in which the enzymes can break up the workload. This is convenient when examining the transition across balance, because varying the parameter \(\eta\) will not shift the point of balance as varying a parameter such as \(\omega\) would. The qualitative behavior state of the system can first be investigated by comparing the total arrival velocity \(\Lambda\) of substrate

\[\Lambda \equiv \lambda_1 + \lambda_2\]
to the total reaction velocity $v_{\text{tot}}$ due to both enzymes:

$$v_{\text{tot}} \equiv \mu_1 X_1 + \mu_2 X_2 = \omega \cdot \left( \frac{(1 - \eta) X_1 + \eta X_2}{\epsilon + (1 - \eta) X_1 + \eta X_2} + \frac{\eta X_1 + (1 - \eta) X_2}{\epsilon + \eta X_1 + (1 - \eta) X_2} \right)$$ (3.45)

Similarly to the arguments in Section 3.3.1, it can be demonstrated that for sufficiently small $\epsilon$, and with $0.5 > \eta \gg \epsilon$, then

$$v_{\text{tot}} \approx \begin{cases} 2\omega, & X_1 > 0 \text{ or } X_2 > 0 \\ 0, & \text{otherwise} \end{cases}$$ (3.46)

Equation 3.46, when true, implies that if either substrate has a nonzero count, the enzymatic network as a whole operates at full speed, such that if one substrate is fully depleted, the remaining substrate is processed at twice the velocity than it otherwise would be. In this limit, the queueing network operates as a symmetric priority queue, where an enzyme may target a lower affinity substrate if its preferred substrate is absent.

Using the above parameterization, we simulated large ensembles of this system stochastically, measuring statistics in the vicinity of the balance point while varying $\eta$ and a production parameter (see Fig. 3.3). Very consistently, transition across the queueing theoretic point of balance leads to qualitative changes in statistics. Strong and concurrent increases in the mean counts of substrate classes are observed when transitioning from an underloaded to an overloaded regime, while peaks in the correlation coefficient between substrate counts occur near the balance point itself.

A feature not originally anticipated is the width of the regime in which correlation between substrate counts remains substantial for exceedingly small $\eta$ (see Fig. 3.3D). This sustained correlation finally becomes small when approximately $\eta < \epsilon$. We believe this wide correlation exists due to the aforementioned priority queue behavior when $\eta$ is small but $\eta \gg \epsilon$, because the system dynamics in this case are very insensitive to the value of $\eta$. A much wider parameter scan to include larger values of $\eta$ supports this conclusion (see Fig. 3.4).
Chapter 3. Proteolytic Crosstalk in Multi-Protease Networks

Figure 3.3: Results for a (2, 2) queueing network. Results for a (2, 2) queueing network (see Section 3.4.2 for details). (A) and (B) show the mean substrate counts for $X_2$ and $X_1$, respectively, as $\lambda_2$ is varied for different values of $\eta$. The balance point ($\lambda_1 + \lambda_2 = 2\omega$, dashed line) indicates the transition between underloaded and overloaded states, and it is observed that increasing the production rate of $X_2$ can strongly induce $X_1$ in the overloaded regime ($\lambda_1 + \lambda_2 > 2\omega$). Only the case $\eta = 0$ (no multi-enzyme crosstalk) failed to exhibit strong induction of $X_1$. (C) The correlation coefficient between $X_1$ and $X_2$ exhibits a strong correlation resonance at the balance point (dashed line) for a wide range of $\eta$ values. (D) Scanning $\eta$ on a log-scale, positive correlation is maintained for very small values of $\eta$. Only when $\eta < \epsilon$ is approximately satisfied does the correlation finally decrease. Parameters are $\lambda_1 = 10$, $\gamma = 0.0231$, $\omega = 10$, and $\epsilon = 10^{-5}$, with $\lambda_2$ and $\eta$ variable.
Chapter 3. Proteolytic Crosstalk in Multi-Protease Networks

Figure 3.4: **Results for a (2, 2) queueing network with varying system-size.** Setting $\lambda_1 = \lambda_2 = \alpha$ and $\omega = \nu \alpha$, the steady state correlation coefficient between $X_1$ and $X_2$ was measured while scanning the coupling parameter $\eta$ and the system-size parameter $\alpha$. This measurement was performed for each combination of three values of $\nu$ and $\epsilon$. Measurement was over an ensemble of 20,000 simulation trajectories for each set of unique parameters. The parameter value $\gamma = 0.0231$ is kept constant across these simulations.

To further investigate this phenomenon, we numerically solved the corresponding master equation of the system on a large grid representing the space of states. We analyzed the system using a non-equilibrium steady state approach detailed by Schnakenberg [96] (see Methods in Section 3.3.5), revealing strong non-equilibrium potentials and currents in the system generated by priority queue behavior near the boundaries of the space of states (see Fig. 3.5). Quantitatively similar results hold for a wide range of small $\eta$ values. As an aside, these non-equilibrium potentials and currents both vanish when either $\eta = 0$ or $\eta = 0.5$ (based on numerical results calculated as above, and based on analytic results in the software package Maple, results not shown), implying reversible systems in these cases.
Figure 3.5: Results for a \((2,2)\) queueing network. A more detailed analysis of a particular \((2,2)\) system with small \(\eta\) (small crosstalk) using the thermodynamic formalism of Schnakenberg [96] (see Section 3.4.2). We plot the steady state (A) probability, (B) cycle current, (C) cycle potential, and (D) cycle power (heat generation) for a system set to balance, with parameters \(\lambda_1 = \lambda_2 = 10\), \(\gamma = 0.0231\), \(\omega = 10\), \(\eta = 0.1\), and \(\epsilon = 10^{-5}\). It is seen that the boundary states (either \(X_1 = 0\) or \(X_2 = 0\)), which were associated with priority queueing behavior in Section 3.4.2, contribute strong cycle potentials that in turn generate non-equilibrium cycle currents. Similar cycle potentials and currents persist even for very small \(\eta\), e.g. \(\eta = 10^{-3}\).

Since natural and synthetic networks are unlikely to exhibit the exact symmetry we assumed in the parameter sets explored above, particularly regarding the assumption that all enzymes have equal reaction velocity \(\omega\) for all substrates, we checked whether the qualitative trends in substrate count statistics were robust to general parameter variation (see Fig. 3.6). These results support that our qualitative results are not particularly sensitive to perturbations in our parameter set, which is consistent with our general discussion in Section 3.4.1. However, the reader is advised that special cases, e.g. those discussed relating to Eqs. 3.16–3.20, may not be consistent with the trends established by the above results.
Figure 3.6: **Results for a (2, 2) queueing network with nonidentical enzymes.** A queueing picture robustly predicts qualitative changes in statistics for a (2, 2) system when \( \omega_1 \) (the maximum reaction velocity of enzyme 1) is not necessarily equal to \( \omega_2 \) (the maximum reaction velocity of enzyme 2). The value of \( \lambda_2 \) was varied while fixing the value \( \lambda_1 = 10 \), thus transitioning the system through different queueing regimes. The values of \( \omega_1 \) and \( \omega_2 \) were independently varied \( \pm 20\% \) from a central value of \( \omega_1 = \omega_2 = 10.0 \), leading to nine unique pairs of values for \( (\omega_1, \omega_2) \). Both (A) the steady state mean substrate count of \( X_1 \) and (B) the steady state correlation between \( X_1 \) and \( X_2 \) exhibit qualitative changes when the system parameters cross the balance point condition \( \lambda_1 + \lambda_2 = \omega_1 + \omega_2 \). Namely, \( X_1 \) is induced in the overloaded regime \( (\lambda_1 + \lambda_2 > \omega_1 + \omega_2) \), and the correlation exhibits a peak near the balance point. Each of the five possible balance point conditions for our choices of \( (\omega_1, \omega_2) \) are represented by vertical dashed black lines. Measurement was over an ensemble of 20,000 simulation trajectories for each set of unique parameters. The parameter values \( \gamma = 0.0231, \eta = 0.1, \text{ and } \epsilon = 10^{-5} \) are kept constant across these simulations. The legend for Panels (A) and (B) is displayed to the right of these panels. Panels (C) and (D) are identical to (A) and (B), but the value \( \epsilon = 1.0 \) was instead used. Results in this case are qualitatively similar to the case \( \epsilon = 10^{-5} \).
Coupled Substrates of a \((3, 2)\) System

A natural extension of the above simplified \((2, 2)\) system includes a third substrate whose production leads to the coupling of two other orthogonal and otherwise noninteracting subsystems. As before, we write the affinities of each substrate for each enzyme as a function of a coupling parameter \(\eta\)

\[
\begin{align*}
\kappa_{11} &= 1 \\
\kappa_{21} &= 0 \\
\kappa_{12} &= 0 \\
\kappa_{22} &= 1 \\
\kappa_{13} &= \eta \\
\kappa_{23} &= 1 - \eta
\end{align*}
\]

Thus, \(X_1\) and \(X_2\) are not directly coupled, but \(X_3\) may couple to \(X_1\) and \(X_2\) and thus indirectly couple \(X_1\) and \(X_2\). Special values for \(\eta\) are as follows: \(\eta = 0\) implies that \(X_3\) is only coupled to \(X_2\), \(\eta = 1\) implies that \(X_3\) is only coupled to \(X_1\), and \(\eta = 0.5\) implies that \(X_3\) is equally coupled to \(X_1\) and \(X_2\). Expressing the enzymatic processing of each substrate as a single reaction, as before, yields the following \((3, 2)\) system

\[
\begin{align*}
&\emptyset \xrightarrow{\lambda} X_j \\
&X_j \xrightarrow{\gamma} \emptyset \\
&X_j \xrightarrow{\mu_j} \emptyset
\end{align*}
\]

for \(j \in \{1, 2, 3\}\), and using the definitions

\[
\begin{align*}
\mu_1 &= \frac{\omega}{\epsilon + X_1 + \eta X_3} \\
\mu_2 &= \frac{\omega}{\epsilon + X_2 + (1 - \eta) X_3} \\
\mu_3 &= \frac{\omega \eta}{\epsilon + X_1 + \eta X_3} + \frac{\omega (1 - \eta)}{\epsilon + X_2 + (1 - \eta) X_3}
\end{align*}
\]

We restrict our investigation of parameter space to \(0 \leq \eta \leq 0.5\), due to symmetry.

As before, we simulated this system stochastically, measuring statistics around the balance point while varying \(\eta\) and a production parameter. Simulation results appear in Fig. 3.7. Results are analogous to the \((2, 2)\) system, where qualitative changes in system statistics are closely related to changing queueing theoretic regimes, and in particular, a significant positive correlation peak robustly arises near the balance point. Unlike the \((2, 2)\) system, however, an additional peak of weak positive correlation exists below the balance point for very small \(\eta\) (see the white dashed line in Fig. 3.7D). We hypothesize that this weak correlation peak may be associated with the balance point of a single enzyme (approximately \(\lambda_3 = 5\) in this case), owing to crosstalk being rather weak, but this is a matter for future investigation.
Figure 3.7: Results for a (3, 2) queueing network. Similar to Fig. 3.3, but for a (3, 2) queueing system (see Section 3.4.2 for details). Coupling between the substrates $X_1$ and $X_2$ is generated indirectly by another substrate $X_3$. As in other cases, induction of mean values and a peak in correlation coefficient are closely related to the point of balance ($\lambda_1 + \lambda_2 + \lambda_3 = 2\omega$, black dashed line). An additional positive peak in correlation occurs for very small $\eta$ near the condition $\lambda_3 = 5$ (white dashed line). Parameters are $\lambda_1 = \lambda_2 = 10$, $\gamma = 0.0231$, $\omega = 15$, and $\epsilon = 10^{-5}$, with $\lambda_3$ and $\eta$ variable.

3.4.3 Numerical Results for Degrade-and-Fire (DF) Oscillators Coupled Through Queueing

An application of multi-enzyme coupling is the queueing-based synchronization of oscillations, as has been recently been investigated experimentally using synthetic gene oscilla-
tors [85]. We leverage a simple modification of our core queueing network model to study the coupling of synthetic gene oscillators via proteolytic activity, i.e. where the substrates are protein repressors (with a delay in their production) and the enzymes are proteases.

**Coupled Goodwin Oscillators of a (2, 2) System**

As described in Methods Section 3.3.2, $X_1$ and $X_2$ in the simple (2, 2) system in Section 3.4.2 were replaced with oscillator models. This mimics two coupled gene oscillators, for example, ClpXP-based and ClpAP-based (indicating the primary protease) oscillators, but where components in each oscillator may be targeted to the “wrong” (lower affinity) protease. Parameterization of substrate affinity by the parameter $\eta$ is as before in Section 3.4.2. We used the correlation coefficient between $X_1$ and $X_2$ as a proxy for synchronization, because this correlation coefficient should become large (close to 1) when the two oscillators exhibit a common waveform.

![Figure 3.8](image)

**Figure 3.8: Results for a (2, 2) oscillator system.** Results for a (2, 2) oscillator system (see Section 3.4.3 for details). Sample trajectories are presented for identical oscillators ($\nu = 1$) either with (A) $\eta = 0.0$ (no crosstalk) or (B) $\eta = 0.5$ (strong crosstalk). Oscillators with no crosstalk show no apparent synchronization, while oscillators with strong crosstalk show nearly perfect synchronization. (C) Further evidence of synchronization is revealed by measuring the steady state correlation between $X_1$ and $X_2$ for a wide range of oscillator parameters. Parameters are $\beta = 2$, $C_0 = 10$, $\alpha_1 = 100$, $\alpha_2 = \nu \alpha_1$, $\gamma = 0.0231$, $\omega = 10$, and $\epsilon = 10^{-5}$, with $\nu$ and $\eta$ variable.
We found that the model oscillators can be robustly synchronized by sharing a common degradation pathway (see Fig. 3.8). For our particular parameters, synchronization was strong for a wide range of crosstalk strength $\eta$. We conclude based off of these results that even for mostly insulated proteolytic pathways (e.g. $\eta = 0.1$), oscillators may strongly synchronize with each other. This conclusion presents a challenge when building orthogonal oscillators within the same cell.

The basic mechanism of synchronization via queueing can be informally argued by considering the degradation phase of two DF oscillators, during which time only proteolytic queueing reactions are relevant. Suppose two DF oscillators initially have different counts for their repressors $X_1$ and $X_2$, and the time of the next firing event for an oscillator is determined by the time when its respective repressor becomes small (near zero) in value. If $X_1$ and $X_2$ are strongly coupled due to sharing or partially sharing a protease, degradation of these proteins occurs in such a way that they approach small values together at the same time. This can be illustrated by considering simple degradation (no production reactions) in a $(2,2)$ system (see Fig. 3.9). Thus, oscillators should then fire at the same time, which leads to synchrony.

![Figure 3.9](image)

Figure 3.9: **Illustration of queueing-based synchronization for a $(2,2)$ system.** Illustration of queueing-based synchronization for a $(2,2)$ system. Shown are the mean values for a $(2,2)$ system as a function of time with initial conditions $X_1 = 200$ and $X_2 = 100$, for either (A) $\eta = 0.0$, (B) $\eta = 0.001$, (C) $\eta = 0.1$, or (D) $\eta = 0.5$. Even for relatively weak crosstalk, the mean values degrade to approximately the same point in time due to the effect of shared protease. If these components were instead oscillators, this effect can be imagined to synchronize the next firing time of the oscillators. Parameters are $\gamma = 0.0231$, $\omega = 10$, and $\epsilon = 10^{-5}$. 
Coupled Goodwin Oscillators of a (3, 2) System

In direct analogy to the previous section, $X_1$ and $X_2$ in the simple (3, 2) system in Section 3.4.2 were replaced with oscillator models. In this case, oscillators are only coupled through the species $X_3$. The degree of crosstalk is encoded by the parameter $\eta$ as in Section 3.4.2.

As illustrated in Fig. 3.10, the degree of apparent synchronization (measured here by correlation) between oscillators is much weaker than in the (2, 2) oscillator model. This result is consistent with the similarly weaker correlation in the simple (3, 2) as compared to the simple (2, 2) system, where the poorer correlation might be explained by the additional degree of separation. Interestingly, unlike the (2, 2) oscillator system, the (3, 2) oscillator system can exhibit weak but statistically meaningful negative correlations, particularly for small $\eta$. We hypothesize that this may due to the oscillators slightly preferring an anti-phase mode of synchronization, but a more complete investigation is left for future work.
Figure 3.10: **Results for a (3, 2) oscillator system.** Results for a (3, 2) oscillator system (see Section 3.4.3 for details). The steady state correlation between oscillator counts $X_1$ and $X_2$ is plotted on both (A) a linear scale and (B) a log scale for $\eta$. Overall, the influence of a third substrate is observed to produce weak positive or negative correlations between the two oscillators. These correlations are sufficiently small that we claim synchronization is weak. Parameters are $\beta = 2$, $C_0 = 10$, $\alpha_1 = 20$, $\alpha_2 = 20$, $\gamma = 0.0231$, $\omega = 10$, and $\epsilon = 10^{-5}$, with $\lambda_3$ and $\eta$ variable.
3.5 Conclusions

In this work, we discussed how shared enzymatic (or proteolytic) activity among substrate classes can lead to strong statistical dependence between the counts of substrate. This basic result for two or more substrates sharing a single enzyme has been noted before, but to our knowledge, our investigation is among the first to discuss whether the result holds in the case of more than one shared enzyme. We conclude based on investigation of simple models (no regulation in the production of substrate) that a theoretical picture based on queueing regimes (underloaded, overloaded, and balanced) provides a meaningful explanation for how the strong statistical dependence observed in these multi-protease order networks can arise. Furthermore, we illustrated that simple oscillator systems can be strongly synchronized by this mechanism of coupling. Overall, this work further supports that queueing can be both an intuitive and quantitatively accurate formalism for enzymatic networks.

These results have extended our understanding of single and multi-enzyme parallel processing systems and should inform the investigation of a number of natural and synthetic systems. One potential application would be in understanding how proteolytic queueing can coordinate toxin-antitoxin (TA) modules. TA modules consist of a gene encoding a stable toxin and an unstable antitoxin that effectively neutralizes the toxin. They are closely associated with bacterial persistence [12,13,59,66,67,110], which is a key factor in chronic infections [67,108]. Unstable antitoxins are typically degraded by one or more proteases, such as the Lon and ClpP proteases [12,13,66,67,103,110]. Preliminary theoretical analysis (data not shown) suggests that queueing of different classes of antitoxin to one or more proteases can strongly synchronize and thus coordinate different TA modules. We predict this effect would complement other proposed coordination mechanisms [31]. Another potential application of our results, if sufficiently generalized, would be for networks of sRNA and mRNA molecules, where a queueing approach has already been proposed to be useful [3,83]. For instance, multiple species of sRNA regulating the degradation of multiple species of mRNA could mimic multiple species of enzyme targeting multiple species of substrate, though degradation of sRNA itself during its regulatory activity is an interesting complication not included in our own analysis.

3.6 Acknowledgements

Funding for this research was provided by the National Science Foundation under Grant No. MCB-1330180.
Chapter 4

Proteolytically Coordinated Activation of Toxin-Antitoxin Modules

4.1 Abstract

Chronic infections present a serious threat to the health of humans by decreasing life expectancy and quality. They have more recently been attributed to the existence of persistor cells within bacterial populations which constitute a small fraction of the population capable of surviving a wide range of environmental stressors including starvation, DNA damage, and heat shock. Persistence also allows the survival of successive applications of antibiotics resulting in chronic infections. Persistence has been strongly linked to toxin-antitoxin (TA) modules, operons with an evolutionarily conserved motif including a toxin that halts cell growth and an antitoxin that under healthy conditions neutralizes the toxin, typically by forming a complex which protects the antitoxin from rapid proteolytic degradation and performs some regulatory action on the operon. While many such modules have been identified and studied in a wide range of organisms, little consideration of interactions between multiple modules within a single host has been made. Moreover, the multitude of different antitoxin species share a limited number of proteolytic pathways, strongly suggesting competition between antitoxins for degradation machinery. Here we present a theoretical understanding of the dynamics of multiple TA modules whose activity is coupled through either proteolytic activity, a toxic effect on cell growth rate, or both. We also present two theoretical mechanisms by which the persistent state is potentially tunable by regulation of proteolysis. Such regulation or indirect coordination between multiple TA modules may be at the heart of bacterial robustness owed to persistence.
4.2 Introduction

It is well established that many bacteria are capable of collective behaviors which yield survival strategies unavailable to individual cells [24, 30, 37]. Bacterial persistence is one such behavior which populations engage in, providing a comprehensive response to deadly events within the environment. Persistence is characterized by a small fraction of a bacterial population occupying a quasi-dormant (persistent) state in which the cell hardly metabolizes or grows and does not divide. This state provides immediate robustness against a wide range of environmental stressors such as starvation and antibiotic treatment, allowing the small subpopulation to survive events which would kill most normally growing cells [5,12,13,30,59,66–68]. With modern medicine, this strategy is indispensable to bacteria which are capable of chronic infection within humans [30,67,108]. It has been claimed that as much as 65% to 80% of all bacterial infections are attributable to bacterial persistence [30,31]. Persistence is also strongly linked to biofilm formation and survival [37,66,67], which is both necessary for many healthy processes in humans and the most likely source of hospital borne disease [24].

A leading candidate for the primary mechanism of bacterial persistence are toxin-antitoxin (TA) modules [12,13,30,31,66–68,95,110]. TA modules are small gene networks following a motif which includes two or more genes within an operon, one of which is a relatively stable toxin which slows the growth of the cell when it accumulates, the other of which is a relatively unstable (readily degraded by ATP-dependent proteases) antitoxin which typically neutralizes the toxin. The antitoxin is capable of forming complexes with its associated toxin, and either the antitoxin or complexes containing it can often bind to the operon negatively regulating transcription [12–14,42,76,110]. This tends to enforce a state where almost all of the antitoxin molecules of the cell are bound to toxin molecules, thereby neutralizing them. This state is in contrast to a high-toxin state where a sufficient quantity of free toxin accumulates, resulting in cell growth arrest characteristic of the persistent state [12,13].

Many bacteria have a surprisingly high number of TA modules (at least 36 within Escherichia coli [30,31,110] and at least 88 within Mycobacterium tuberculosis [30,31,37,68,76,95]), though why this is the case has not been well explained [30]. The nonpathogenic organism Mycobacterium smegmatis has only two putative TA modules, suggesting that the plurality of TA modules is related to the pathogenicity of some bacterial strains, or that many TA modules play a role in achieving the particularly long periods of dormancy associated with M. tuberculosis [30,110]. It is thus important for human medicine to understand the interplay of potentially many TA modules in promoting pathogenicity and in particular bacterial persistence. For E. coli, it has been shown that removal of either the Lon protease or a number of known TA modules is sufficient to strongly suppress persistence [67,68]. This suggests that both the presence of specific TA modules and proteolytic degradation are necessary for persister formation. In E. coli, there are three proteases (Lon, ClpAP, and ClpXP) responsible for more than 70% of ATP-dependent protein degradation within the cell [60,61]. In particular Lon is responsible for approximately 50% of defective protein degradation and likely degrades the antitoxins of more than 10 TA modules in E. coli [67,68]. There are thus
far fewer proteolytic pathways than the number of TA modules, implying that likely more than one TA module is actively utilizing a given proteolytic pathway. This may result in competition between antitoxin proteins for proteolytic machinery which leads to processing bottlenecks under specific conditions.

Bottlenecks in proteolytic processing have a coherent explanation in the context of queueing theory. Queueing theory is a formalism used to describe the dynamics of a set of servers which perform a set of tasks for a set of customers. By drawing an analogy between servers and proteases, and customers and the degradation of proteolytic targets, queueing theory offers insight into the effects of bottlenecks. In particular queueing theory depicts several regimes characterized by drastic differences in the statistics associated with the completion of tasks. The underloaded regime is marked by rapid processing of tasks upon arrival, occasionally unoccupied servers, and a lack of waiting lines (queues). When the servers are overwhelmed with customers, queues form as the maximum processing rate of the servers is insufficient to keep up with the influx of customers. When a system transitions between these two regimes it is said to occupy the balanced regime [18,44,50,71,78]. Previous work describing bottlenecks in enzymatic processing using queueing theory have identified a phenomenon by which the molecular counts of degradation targets that compete for proteolytic machinery are correlated as the system transitions into the balanced and overloaded regimes. Furthermore, these correlations are maximized at the queueing theoretic point of balance. This phenomenon is known as correlation resonance [18,71,78]. It is conceivable that proteolytic competition between TA modules provides a mechanism by which they might coordinate their effects, perhaps in a relevant way for promoting persistence and pathogenicity in general.

While several investigations have considered the dynamics of a single TA module [12,13], more recently, work has shifted to explore the interactions of multiple TA modules [30,31,42]. This chapter presents several models containing one or two TA module components represented by chemical reactions based on the well-studied TA module \textit{mazEF} [28,37,58,76,95,110]. Using extensive stochastic simulation, measurements of toxic event statistics were made for the purpose of describing the effect of proteolytic competition between TA modules on persistence. In living cells, TA modules experience the same effective dilution rate associated with cell growth and division, while accumulation of any class of many distinct toxins may affect the growth rate. Through this effect, multiple TA modules in a single cell are always coupled [30,31]; however, the role of this coupling is investigated by providing an artificial independence of growth rates experienced by distinct TA modules. This representation enables a distinction between effects of either mode of coupling on the statistics of persistence attributable to toxic events.

Regulation of the persister fraction within a colony should depend directly on regulation of toxic state entry and exit of individual cells. It is thought that toxic state entry is at least partially a stochastic process [37,66,67,108]. While the population fraction of persister cells is typically within the range of 1 in $10^4$ to 1 in $10^6$ (for wild type \textit{E. coli}) [30,59,66,67], this fraction generally depends on the environment and population maturity [5,37,66]. The molecule ppGpp, whose accumulation is correlated with environmental stress such as
starvation \[15, 60, 62, 66, 67, 98, 108\], promotes the accumulation of the molecule inorganic polyphosphate (polyP) \[60, 66, 67, 103\]. PolyP interacts with the Lon protease, stimulating its processing \[61, 66, 67, 103\]. Several functions have been suggested for polyP stimulated Lon, including in particular the direction of ribosomal proteins to Lon for degradation and the heightened degradation of many TA module antitoxins \[42, 61, 66, 67, 103\]. It has been shown experimentally in \textit{E. coli} that strains which do and do not readily accumulate polyP have significantly increased and reduced persister fractions respectively. Moreover, heightened persister fractions with strains which readily accumulate polyP but lack either Lon or a number of TA modules is not observed, indicating that the ability to accumulate polyP correlates with persistence mediated by both Lon and many TA modules \[67\]. The TA module on which the models in this chapter are based contains a third protein (MazG) which is neither the toxin (MazF) nor the antitoxin (MazE) \[62\]. MazG activation reduces the concentration of ppGpp \[62\] which helps return the activity of Lon proteases within the cell to their basal state. As it is known that the activity of the Lon protease is regulated by these components and possibly others \[103\], this chapter presents two simple mechanisms involving proteolytic regulation by which the toxic events of a TA module could be tuned.

### 4.3 Methods

We consider systems containing one or more identical TA modules which are based on a well-studied TA module as found in \textit{E. coli}. Specifically, the stoichiometric ratio of the complex formed by the toxin and antitoxin and the mechanism of toxicity where the toxin cleaves mRNAs with specific sequences are based on \textit{mazEF} \[28, 37, 58, 76, 110\]. \textit{mazEF} is of interest for exploring proteolytic coupling as the antitoxin MazE is degraded by both ClpAP and Lon \[28, 37, 42, 110\], and could thus even indirectly couple additional modules as in \[78\]. The toxin MazF inhibits the translation of approximately 90% of all proteins in \textit{E. coli} \[76\] and thus plays an important role in specifying protein production during environmental stress. The only mRNAs in the model are those of the toxin and antitoxin, which have nine and two cut sites respectively for the \textit{mazEF} module. A model with one TA module is presented in section 4.3.1 (see Figure 4.1 for a schematic representation). We then present a model with two modules which are coupled through an effect of toxin on growth rate and by sharing of proteolytic pathways in section 4.3.2. Both modes of coupling are parameterized such that the dependence of toxin event statistics on each mode of coupling can be explored. We then present two potential mechanisms which offer regulation on the duration of toxin events in section 4.3.3. Finally, a link to the software used is provided in section 4.3.4, the algorithm for event detection is described in section 4.3.5, and measurement of event statistics is described in section 4.3.6.
Figure 4.1: Schematic of single TA module model based on \textit{mazEF} in \textit{E. coli}. A single mRNA is produced encoding both a toxin and an antitoxin. The toxin forms a dimer, which then forms a complex with the antitoxin. This complex also forms a dimer. The antitoxin, complex, and complex dimer all repress transcription of the mRNA (red lines). Proteases can degrade free antitoxin and antitoxin bound in complexes (blue lines). Free toxin and its dimer inhibit translation of both the toxin and antitoxin by cleaving the mRNA encoding them (orange lines). All species are also subject to effective dilution by cell growth and division (not shown). The rate of dilution is dependent on the level of free toxin and its dimer because cell growth is slowed by the accumulation of translation inhibiting toxins.

4.3.1 Single TA Module Model

The model of a single TA module based on \textit{mazEF} in \textit{E. coli} contains the toxin and antitoxin mRNAs (\(t\) and \(a\) respectively), the toxin and antitoxin proteins (\(T\) and \(A\) respectively), a complex formed by the toxin and antitoxin (\(C\)), and a dimer of both the toxin and the complex. Both mRNAs are produced by a single reaction as one mRNA is produced which encodes both MazE and MazF in the \textit{mazEF} module found in \textit{E. coli}. As this mRNA could potentially be cleaved in such a way that only either MazE or MazF could be translated from it, and because \textit{mazE} and \textit{mazF} have very different numbers of MazF specific cut site sequences, the mRNAs are modeled independently. The functional form of the transcription rate \(\lambda_{ta}\) contains autorepression of the module. Translation of both toxin and antitoxin mRNAs is represented by first order reactions with constant rates. Complex formation results in a stoichiometric ratio of two toxins to one antitoxin as observed in the \textit{mazEF}
module [28, 110]. Both toxin and antitoxin mRNAs are cleaved at a rate proportional to the sum of the toxin and its dimer. The degradation rates of free antitoxin and antitoxin sequestering toxin in the form of complexes is represented using functional forms similar to those used in [71, 78]. All species (“Z” in the following equations) are diluted at rate Γ, representing the process of cell growth and division. The functional form of Γ represents the effect of the toxin on growth rate as it decreases as either the toxin or its dimer increases. The reactions used for the single TA module model are as follows:

\[
\begin{align*}
\varnothing & \xrightarrow{\lambda_{\varnothing}} \text{a} + \text{t} \quad (4.1) \\
\text{a} & \xrightarrow{\alpha_a} \text{a} + \text{A} \quad (4.2) \\
\text{t} & \xrightarrow{\alpha_t} \text{t} + \text{T} \quad (4.3) \\
2\text{T} & \xrightarrow{k_C} \text{T}_2 \quad (4.4) \\
\text{T}_2 + \text{A} & \xrightarrow{k_C} \text{C} \quad (4.5) \\
2\text{C} & \xrightarrow{k_C} \text{C}_2 \quad (4.6) \\
\text{t} & \xrightarrow{\beta_t} \varnothing \quad (4.7) \\
\text{a} & \xrightarrow{\beta_a} \varnothing \quad (4.8) \\
\text{A} & \xrightarrow{\zeta} \varnothing \quad (4.9) \\
\text{C} & \xrightarrow{\zeta} \text{T}_2 \quad (4.10) \\
\text{C}_2 & \xrightarrow{\zeta} \text{T}_2 + \text{C} \quad (4.11) \\
\text{Z} & \xrightarrow{\Gamma} \varnothing \quad (4.12)
\end{align*}
\]

\[
\begin{align*}
\lambda_{\varnothing} &= \frac{\sigma}{1 + k_0 (\text{A} + \text{C} + \text{C}_2)} \quad (4.13) \\
\alpha_a &= \nu \alpha_t \quad (4.14) \\
\beta_t &= \beta_T (\text{T} + \text{T}_2) \quad (4.15) \\
\beta_a &= \omega \beta_t \quad (4.16) \\
\zeta &= \frac{\mu}{\epsilon + \text{A} + \text{C} + \text{C}_2} \quad (4.17) \\
\Gamma &= \frac{\gamma}{1 + \beta_y (\text{T} + \text{T}_2)} \quad (4.18)
\end{align*}
\]
The parameter $\nu$ dictates the relative translation rate of the antitoxin mRNA to the toxin mRNA. The value 1.25 was chosen because the antitoxin is typically produced faster than the toxin [37]. The parameter $\omega$ encodes the relative cleavage rate of the antitoxin mRNA to the toxin mRNA by the toxin $T$. The value 0.2 was chosen because there are nine cut sites for the toxin MazF within the $mazF$ mRNA and two cut sites within the $mazE$ mRNA (in $E. coli$). The parameter $\beta_T$ encodes the strength of the toxic effect on translation as it represents the cleavage rate of mRNAs in the system by the toxin $T$. The parameter $\beta_g$ encodes the strength of the effect of the toxin on growth rate. The maximum dilution rate $\gamma$ was set to 1.0. Transcription ($\sigma$), translation ($\alpha_t$), complex formation ($k_C$), and degradation ($\mu$) are faster processes all represented with rates far greater than $\gamma$. $k_0$ encodes the strength of autorepression of the module by the antitoxin $A$, the complex $C$, and the complex dimer $C_2$. $\epsilon$ is related to the affinity of the protease for an unbound state (it is the ratio of the unbinding to binding rates between protease and its target [71]), and is thus small relative to the affinity of the degradation targets as seen in [78].

### 4.3.2 Double TA Module Model

In the interest of exploring coupling of multiple TA modules via shared proteolytic pathways and by sharing a growth rate dependent on toxic effects, we include a double module model, which contains two identical modules to the model in Section 4.3.1 (see Figure 4.2). Two proteolytic pathways are used so that the modules may potentially be proteolytically uncoupled. The forms of $\zeta_1$ and $\zeta_2$ are similar to those seen in [78]. In a real cell, two TA modules could never experience different growth rates. The forms of $\Gamma_1$ and $\Gamma_2$, which are similar to the forms of $\zeta_1$ and $\zeta_2$, artificially allow the modules to experience different growth rates. The strength of proteolytic coupling and growth rate coupling are encoded in the
parameters $\eta_p$ and $\eta_g$ respectively. These additional parameters are bounded below by 0.0 where coupling is nonexistent and 0.5 representing maximal coupling. As either parameter is increased above 0.5, the identities of the associated pathways (proteolytic or dilutive) are effectively exchanged due to symmetry in the representations of $\zeta_1$, $\zeta_2$, $\Gamma_1$, and $\Gamma_2$. The reactions used for the double TA module model are as follows:
\[ \emptyset \xrightarrow{\lambda a_1} a_1 + t_1 \]  \hspace{1cm} (4.30)

\[ \emptyset \xrightarrow{\lambda a_2} a_2 + t_2 \]  \hspace{1cm} (4.31)

\[ a_1 \xrightarrow{\alpha a} a_1 + A_1 \]  \hspace{1cm} (4.32)

\[ t_1 \xrightarrow{\alpha t} t_1 + T_1 \]  \hspace{1cm} (4.33)

\[ a_2 \xrightarrow{\alpha a} a_2 + A_2 \]  \hspace{1cm} (4.34)

\[ t_2 \xrightarrow{\alpha t} t_2 + T_2 \]  \hspace{1cm} (4.35)

\[ 2T_1 \xrightarrow{kC} T_1 \]  \hspace{1cm} (4.36)

\[ T_1 + A_1 \xrightarrow{kC} C_1 \]  \hspace{1cm} (4.37)

\[ 2C_1 \xrightarrow{kC} C_1 \]  \hspace{1cm} (4.38)

\[ 2T_2 \xrightarrow{kC} T_2 \]  \hspace{1cm} (4.39)

\[ T_2 + A_2 \xrightarrow{kC} C_2 \]  \hspace{1cm} (4.40)

\[ 2C_2 \xrightarrow{kC} C_2 \]  \hspace{1cm} (4.41)

\[ t_1 \xrightarrow{\beta t_1} \emptyset \]  \hspace{1cm} (4.42)

\[ a_1 \xrightarrow{\beta a_1} \emptyset \]  \hspace{1cm} (4.43)

\[ t_2 \xrightarrow{\beta t_2} \emptyset \]  \hspace{1cm} (4.44)

\[ a_2 \xrightarrow{\beta a_2} \emptyset \]  \hspace{1cm} (4.45)

\[ A_1 \xrightarrow{\zeta_1} \emptyset \]  \hspace{1cm} (4.46)

\[ C_1 \xrightarrow{\zeta_1} T_1 \]  \hspace{1cm} (4.47)

\[ C_1 \xrightarrow{\zeta_1} T_1 + C_1 \]  \hspace{1cm} (4.48)

\[ A_2 \xrightarrow{\zeta_2} \emptyset \]  \hspace{1cm} (4.49)

\[ C_2 \xrightarrow{\zeta_2} T_2 \]  \hspace{1cm} (4.50)

\[ C_2 \xrightarrow{\zeta_2} T_2 + C_2 \]  \hspace{1cm} (4.51)

\[ Z_1 \xrightarrow{\Gamma_1} \emptyset \]  \hspace{1cm} (4.52)

\[ Z_2 \xrightarrow{\Gamma_2} \emptyset \]  \hspace{1cm} (4.53)
\[ \lambda_{ta1} = \frac{\sigma}{1 + k_0 (A_1 + C_1 + C_1^2)} \] (4.54)

\[ \lambda_{ta2} = \frac{\sigma}{1 + k_0 (A_2 + C_2 + C_2^2)} \] (4.55)

\[ \alpha_a = \nu \alpha_t \] (4.56)

\[ \beta_{t1} = \beta_T (T_1 + T_1^2) \] (4.57)

\[ \beta_{a1} = \omega \beta_{t1} \] (4.58)

\[ \beta_{t2} = \beta_T (T_2 + T_2^2) \] (4.59)

\[ \beta_{a2} = \omega \beta_{t2} \] (4.60)

\[ \zeta_1 = \frac{(1 - \eta_p) \mu}{\epsilon + (1 - \eta_p) (A_1 + C_1 + C_1^2) + \eta_p (A_2 + C_2 + C_2^2)} + \frac{\eta_p \mu}{\epsilon + \eta_p (A_1 + C_1 + C_1^2) + (1 - \eta_p) (A_2 + C_2 + C_2^2)} \] (4.61)

\[ \zeta_2 = \frac{(1 - \eta_g) \gamma}{1 + (1 - \eta_g) \beta_g (T_1 + T_1^2) + \eta_g \beta_g (T_2 + T_2^2)} + \frac{\eta_g \gamma}{1 + \eta_g \beta_g (T_1 + T_1^2) + (1 - \eta_g) \beta_g (T_2 + T_2^2)} \] (4.62)

\[ \Gamma_1 = \frac{(1 - \eta_g) \gamma}{1 + (1 - \eta_g) \beta_g (T_1 + T_1^2) + \eta_g \beta_g (T_2 + T_2^2)} + \frac{\eta_g \gamma}{1 + \eta_g \beta_g (T_1 + T_1^2) + (1 - \eta_g) \beta_g (T_2 + T_2^2)} \] (4.63)

\[ \Gamma_2 = \frac{(1 - \eta_g) \gamma}{1 + (1 - \eta_g) \beta_g (T_1 + T_1^2) + \eta_g \beta_g (T_2 + T_2^2)} + \frac{\eta_g \gamma}{1 + \eta_g \beta_g (T_1 + T_1^2) + (1 - \eta_g) \beta_g (T_2 + T_2^2)} \] (4.64)
4.3.3 Single TA Module Return Mechanisms

We present two possible mechanisms to offer regulation on the time spent in a persistent state in the context of a single TA module model. We permit the affinity of the complex and its dimer to differ from that of the antitoxin and include the degradation of free toxin and its dimer also with a different affinity than that of the antitoxin. With these modifications and the values of $\kappa_C$ and $\kappa_T$ as below, this constitutes an equivalent model to the previously mentioned single TA module model. The affected reactions are as follows:

\[
\begin{align*}
    A & \xrightarrow{\zeta_A} \emptyset \quad (4.65) \\
    C & \xrightarrow{\zeta_C} T_2 \quad (4.66) \\
    C_2 & \xrightarrow{\zeta_C} T_2 + C \quad (4.67) \\
    T & \xrightarrow{\zeta_T} \emptyset \quad (4.68) \\
    T_2 & \xrightarrow{\zeta_T} T \quad (4.69) \\
    \zeta_A = & \frac{\mu}{\epsilon + A + \kappa_C (C + C_2) + \kappa_T (T + T_2)} \quad (4.70) \\
    \zeta_C = & \zeta_A \kappa_C \quad (4.71) \\
    \zeta_T = & \zeta_A \kappa_T \quad (4.72) \\
    \kappa_C = & 1.0 \quad (4.73) \\
    \kappa_T = & 0.0 \quad (4.74)
\end{align*}
\]

We then introduce an additional protein $X$ with mRNA $x$. The mRNA $x$, which is only produced in the presence of toxin, is transcribed at rate $\sigma_x$ and translated at rate $\alpha_X$. Both $x$ and $X$ are diluted at rate $\Gamma$. This requires four additional reactions:

\[
\begin{align*}
    \emptyset & \xrightarrow{\sigma_x} x \quad (4.75) \\
    x & \xrightarrow{\alpha_X} x + X \quad (4.76) \\
    x & \xrightarrow{\Gamma} \emptyset \quad (4.77) \\
    X & \xrightarrow{\Gamma} \emptyset \quad (4.78) \\
    \sigma_x = & \rho_x (T + T_2) \quad (4.79) \\
    \rho_x = & 0.1 \quad (4.80) \\
    \alpha_X = & 0.1 \quad (4.81)
\end{align*}
\]
I: Destabilization of Toxin

The protein $X$ permits the degradation of toxin by increasing the affinity $\kappa_T$, while $\beta_X$ encodes the strength of this effect.

$$\kappa_T = \beta_X X \quad (4.82)$$
$$\beta_X = 0.00001 \quad (4.83)$$

II: Stabilization of Complex

The protein $X$ limits the degradation of antitoxin bound to toxin by decreasing the affinity $\kappa_C$, while $\beta_X$ encodes the strength of this effect.

$$\kappa_C = \frac{1}{1 + \beta_X X} \quad (4.84)$$
$$\beta_X = 0.00001 \quad (4.85)$$

4.3.4 Simulation Algorithm

All simulations were performed using a custom implementation of the Gillespie algorithm, run either on a personal computer or a computing cluster. These simulations were executed and analyzed using our custom python package, with a recent version available at www.github.com/ctogle/modular.

4.3.5 Event Detection

Event detection is performed on the total toxin ($T + T_2$) at steady state (last 90% of the realization) for each TA module in the model. The algorithm begins by identifying the low transition threshold ($t_{low}$) and the high transition threshold ($t_{high}$). Given two user specified parameters and a module with toxin $T$, these thresholds are given by the following, which proved to be robust for event detection of the above models:
Chapter 4. Proteolytically Coordinated Activation of Toxin-Antitoxin Modules

\[ z_0 = 0.2 \]  
\[ w_0 = 0.125 \]  
\[ z = \max(T + T_2) \cdot z_0 \]  
\[ w = \max(T + T_2) \cdot w_0 \]  
\[ t_{\text{low}} = \max(1, z - w) \]  
\[ t_{\text{high}} = \min(\max(T + T_2) - 1, z + w) \]

Each data point of total toxin is then considered sequentially to identify transitions between the high and low toxin states. The toxin state is ambiguous until the first point below \( t_{\text{low}} \) is found, at which point transitions can be recorded. This possibly ignores one event which might be occurring when the system reaches 10% of the final simulation time, which is safer than using an event which began before steady state was reached. While in the high toxin state, a low transition is identified when the total toxin count decreases below \( t_{\text{low}} \). While in the low toxin state, a high transition is identified when the total toxin count exceeds \( t_{\text{high}} \). As the toxin state must have been low some time prior to when any high transition threshold is identified, the actual transition is recorded as the last time the total toxin count exceeded \( t_{\text{low}} \) prior to exceeding \( t_{\text{high}} \). If the set of transitions ends with a transition to the high toxin state, this last transition is neglected as the event must have been occurring at the end of the realization. This creates a set of pairs of upward and downward transitions which bound the set of events found within the realization. By construction, each event begins and ends with a data point below \( t_{\text{low}} \), and each event contains at least one data point above \( t_{\text{high}} \).

The set of resulting events is filtered by merging or neglecting events according to the following rules which depend on the user specified parameter \( n_t \):

- Any event containing fewer than \( n_t \) data points is neglected.
- If the preceding event is separated by fewer than \( n_t \) data points, the events are merged.
- If the mean total toxin between two events is above \( t_{\text{high}} \), the events are merged.
- The first event must begin no fewer than \( n_t \) data points after steady state has been reached. The first event is thrown out if this is not the case.
- The last event must end no fewer than \( n_t \) data points before the realization ends. The last event is thrown out if this is not the case.

By constraining the simulation data so that the state of the system is represented every 1 time units and fixing \( n_t = 5 \), we have implicitly ignored any event whose duration is shorter than 5 time units. The resulting set of events satisfies the following criteria.
• Each event begins and ends with a data point below $t_{low}$.
• Each event contains at least one data point above $t_{high}$.
• Each event contains at least $n_t$ data points.
• There are at least $n_t$ data points between any two events which are not part of an event.
• The first event began at least $n_t$ time units after steady state was reached.
• The last event ended at least $n_t$ time units before the end of the realization.

4.3.6 Event Measurement

The widths of all detected events from a single realization are treated together to compute the mean, standard deviation, minimum, maximum, and variance of event widths. The total toxin count data points occurring during any event of the realization are treated together to compute the mean, standard deviation, minimum, maximum, and variance of event heights. This effectively weights the statistics of event height by the widths of the events, as longer events contribute more data points. The probability of being in the high toxin state ($P_{high}$) for a particular module is computed as the sum of all event widths of the module divided by the total time considered. This introduces a small error where events occurring as the system reached steady state or when the realization ended were neglected. As the duration of the realization is far longer than the maximum event width detected, no correction is applied for this source of error. The probability of being in the low toxin state is computed as $1 - P_{high}$. In addition to these statistics, which are computed independently for each TA module of the model being explored, the covariance and correlation of total toxin count of the modules is computed using all data points occurring during any event of one module.

4.4 Results and Discussion

Extensive stochastic simulation of the models in sections 4.3.1, 4.3.2, and 4.3.3 were performed using an implementation of the Gillespie algorithm (see Section 4.3.4) so that steady state measurements of toxic events could be made (see sections 4.3.5 and 4.3.6). For all presented results, event measurements were performed using the last 90% of a single realization of length 1,000,000 time units. The number of detected events is recorded to serve as a proxy for the validity of event statistics.
4.4.1 Single TA Module Model

Measurements of toxic events were made of the single TA module model while varying the translation rate $\alpha_t$ and the complex binding rate $k_C$. Simulation of the single TA module model exhibited bistability for a range of parameters (see figures 4.4, 4.5, and 4.3). Increasing either $\alpha_t$ or $k_C$ increases the probability of being in a toxic state at steady state. While the mean count of toxin during toxic events increased with $\alpha_t$, there was far less of a dependence on $k_C$. These measurements are present in Figure 4.4.

Figure 4.3: Event detection robustness. Event detection of total toxin ($T + T_2$) is demonstrated for the four corners of the parameter space explored with the single TA module model. The process of event detection requires identification of the high and low transition thresholds shown with dashed lines in each plot (see Section 4.3.5 for a detailed description of event detection). Filtered events are marked with red lines and square end points whose positions coincide with the transitions in and out of the toxic state. Event detection depends on three parameters ($z_0 = 0.2, w_0 = 0.125,$ and $n_t = 5$) which robustly identified events for the considered parameter space. In addition to statistics of toxic events, the count of events found is also recorded (shown in the bottom plot for the single module model) to serve as a proxy for the quality of the measured statistics. Parameters are $\sigma = 20, k_0 = 0.05, \nu = 1.25, \omega = 0.2, \beta_T = 5, \beta_g = 0.25, \mu = 100, \epsilon = 0.05,$ and $\gamma = 1,$ while $\alpha_t$ and $k_C$ are varied.

Increasing either $\alpha_t$ or $k_C$ increases the mean duration of toxic events, though the dependence on $\alpha_t$ is stronger. The standard deviation of toxic event duration depends similarly to the mean on $\alpha_t$ but does not appear to depend as strongly on $k_C$. The standard deviation of toxin
Figure 4.4: Measurement of a single TA module model. Event detection was performed on the total toxin ($T + T_2$) permitting characterization of the events generated by the model of a single $mazEF$-like module. Events were measured from a single long trajectory of length 1,000,000 time units for each distinct set of parameter values. For the range of parameters scanned, the lowest event count was more than 450 events (see Fig. 4.3). The bottom left plot demonstrates strong dependence of the steady state probability of being in the high toxin state and the rates of translation ($\alpha_t$) and of complex formation ($k_C$). The bottom right plot shows strong dependence of the mean total toxin count during high toxin events on the translation rate, while there is little dependence on the binding rate. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, and $\gamma = 1$, while the translation rate $\alpha_t$ and the binding rate $k_C$ are scanned.

Count during toxic events does not strongly depend on either $\alpha_t$ or $k_C$. These measurements are present in Figure 4.5. Also apparent in Figure 4.5, the mean and standard deviation of toxic event duration where $\alpha_t = 100$ and $k_C = 100$ are such that some events in the distribution will have a width below the minimum detectable width of 5 time units. The maximum dilution rate $\gamma$ is related to the maximal cell growth rate, and thus corresponds to a doubling time of 20 minutes for fast growing $E. coli$ cells [97]. The value $\gamma = 1$ corresponds to a doubling time of $\ln (2) = 0.693$ time units, thus the minimum event duration of 5 time units corresponds to 7.2 doubling times, or 2.4 hours. This criterion for toxic events is subjective but perhaps reasonable as $E. coli$ persister cells may remain in the toxic state for approximately 14 hours [5]. For further exploration of toxic events, $\alpha_t$ and $k_C$ were fixed at 200 and 1000 respectively.
Chapter 4. Proteolytically Coordinated Activation of Toxin-Antitoxin Modules

Figure 4.5: Characterization of single TA module toxic events. Using event detection and measurement of the total toxin \((T + T_2)\), the mean and standard deviation of both toxic event width and height are measured. Events were collected from a single long trajectory of length 1,000,000 time units for each distinct set of parameter values. For the range of parameters scanned, the lowest event count was more than 450 events (see Fig. 4.3). The top left plot shows strong dependence of mean toxic event width on the translation rate \(\alpha_t\) and relatively weak dependence on the rate of complex formation \(k_C\). The top right plot shows strong dependence of toxic event width standard deviation on \(\alpha_t\) and almost no dependence on \(k_C\). The bottom left plot shows strong dependence of mean toxic event height on \(\alpha_t\) and relatively weak dependence on \(k_C\). The bottom right plot shows weak dependence of toxic event height standard deviation on \(\alpha_t\) and \(k_C\). Parameters are \(\sigma = 20, k_0 = 0.05, \nu = 1.25, \omega = 0.2, \beta_T = 5, \beta_g = 0.25, \mu = 100, \epsilon = 0.05,\) and \(\gamma = 1,\) while \(\alpha_t\) and \(k_C\) are scanned.
4.4.2 Double TA Module Model

The dependence of toxic event statistics on coupling through either proteolytic competition, a toxic effect on cell growth rate, or both was measured using the model in Section 4.3.2 while varying the coupling parameters $\eta_p$ and $\eta_g$. Figure 4.6 contains realizations with four combinations of the two coupling mechanisms considered in this chapter. Without either form of coupling, the modules act independently featuring toxic events as seen in Fig. 4.3. With only coupling through the toxic effect on growth rate, toxic events are shortened, and toxic state entry and exit appear uncorrelated between modules. With maximal proteolytic coupling, toxic event entry and exit is strongly correlated between modules, while toxic event duration is greatly dilated (see Fig. 4.7). The probability of being in the toxic state at steady state depends strongly on $\eta_p$ without growth rate coupling, but appears somewhat independent if growth rate coupling is present (see Fig. 4.7).

![Figure 4.6: Measurement of a double TA module model.](image)

Coupling of identical *mazEF*-like modules via shared proteolytic pathways and a global effect on growth rate is explored using the model described in Section 4.3.2. The event detection process is run on the total toxin count of each module ($T_1 + T_{12}$ and $T_2 + T_{22}$), and additional statistics such as the correlation of toxins are calculated by considering the total toxin counts during the events of one module (see Section 4.3.6 for more details). Proteolytic coupling strongly dilates the width of toxic events and correlates the entry and exit of the toxic state for each module. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\eta_p$ and $\eta_g$ are varied.
Figure 4.7: **Effect of coupling on toxic event probability and correlation.** Coupling of identical *mazEF*-like modules via shared proteolytic pathways and a global effect on growth rate is explored using the model described in Section 4.3.2. Without either growth rate or proteolytic coupling, toxic events are effectively uncorrelated. Proteolytic coupling increases the steady state toxic state probability by dilating and correlating the toxic events of each module. Growth rate coupling lowers the toxic state probability while increasing toxin correlations. Both of these effects are observed for low values of the coupling parameters $\eta_p$ and $\eta_g$. In particular, there is little effect on toxic state probability or toxin count correlation during toxic events when $\eta_p$ is increased above 0.1 (20% of the maximal extent of proteolytic coupling). It should be noted that identical TA modules contained in the same cell may only experience the condition $\eta_g = 0.5$. As the modules are identical, the top left and top right plots are interchangeable as are the bottom left and bottom right plots. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\eta_p$ and $\eta_g$ are varied.

The mean event width in Figure 4.8 shows that event dilation is relatively small, if present at all, without proteolytic coupling. It appears that once proteolytic coupling is present, dilation increases with growth rate coupling, though the data is perhaps too noisy to draw this conclusion for low $\eta_p$. This is likely because once events are strongly dilated, far fewer may be recorded during the 1,000,000 time units which are simulated for each set of parameters (see Figure 4.6). The mean event height appears to have very little dependence on either $\eta_p$ or $\eta_g$ if $\eta_p$ is increased above approximately 0.1 (20% of maximal coupling). For $\eta_p$ below this
value, it appears that mean event width increases with either mode of coupling. Similarly to
the mean event height, the standard deviation of event height depends only weakly on either
$\eta_p$ or $\eta_g$ if $\eta_p > 0.1$. It appears that both modes of coupling reduce the standard
deviation of event height.

Figure 4.8: **Effect of coupling on toxic event width and height.** Coupling of identical
*mazEF*-like modules via shared proteolytic pathways and a global effect on growth rate is
explored using the model described in Section 4.3.2. The mean (top left) and standard devi-
ation (top right) of toxic event widths increases with proteolytic and growth rate coupling.
The mean toxin count during toxic events (bottom left) increases when either form of cou-
pling is present but does not strongly depend on the extent of the coupling. The standard
deviation of toxin count during toxic events (bottom right) decreases when either form of
coupling is increased, showing little dependence above a relatively low level of coupling (20% 
of the maximal extent of proteolytic coupling). Statistics were measured using single long
trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$,
$\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\eta_p$
and $\eta_g$ are varied.
4.4.3 Return Mechanism I

For return mechanism I, identical measurements to those in Section 4.4.1 were made while varying the three parameters which regulate the return mechanisms ($\rho_X$, $\alpha_X$, and $\beta_X$). Event measurements were taken of the mRNA $x$ and protein X as well as the total toxin ($T + T_2$). Figure 4.9 contains realizations of the model with several extreme parameter values. It is apparent that the mean event heights of the mRNA $x$ and protein X can be adjusted relative to the mean event height of total toxin. The strength of the return mechanism (proportional to $\beta_X$) readily tunes the duration of toxin events for both of these situations as desired. Figure 4.9 also contains several measurements of the event counts including the minimum number of events measured for return mechanism I (160).

For the six values of $\beta_X$ measured, the probability of being in the toxic state at steady state is tunable between an approximate maximum value of 0.6 and an approximate minimum value of 0.05. The tunability is weakest at the extreme values of $\beta_X$ where the probability depends most weakly upon the transcription and translation rates ($\rho_X$ and $\alpha_X$ respectively). The dependence, and thus tunability, is maximized where $\beta_X$ is between the extreme values measured. The dependence of the mean event width upon the return mechanism parameters appears to be very similar to that of the toxic state probability (see Fig. 4.11). Mean event height depends more gradually upon $\beta_X$, where peak tunability appears to be found at the maximal considered value $\beta_X = 10^{-5}$ (see Fig. 4.12).
Figure 4.9: **Effect of return mechanism I on toxic events.** Return mechanism I involves three parameters. $\rho_X$ and $\alpha_X$ are the transcription and translation rates of the additional species $X$, and $\beta_X$ regulates the affinity of the protease for the toxin and its dimer. The top left plot shows that with fast production of the species $X$ ($\rho_X = 1.0$ and $\alpha_X = 1.0$) the proposed return mechanism can strongly reduce the duration of toxic events. The bottom left plot shows that with far lower $\beta_X$, events are no longer stifled and the accumulation of $X$ far outpaces that of the toxin. With far lower production parameter values, the top right and bottom right plots show that $\beta_X$ can regulate the width of toxic events with relatively low accumulation of the additional species $X$. The count of events used for statistics are also shown for several values of $\beta_X$, where the minimum event count measured was 160. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
Figure 4.10: Effect of return mechanism I on event probability. A measurement of steady state toxic state probability as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
Figure 4.11: **Effect of return mechanism I on event width.** A measurement of the mean toxic event width as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
Figure 4.12: **Effect of return mechanism I on event height.** A measurement of the mean toxin count during toxic events as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
4.4.4 Return Mechanism II

Return mechanism II was measured identically to return mechanism I, with the exception that the region of interest of $\beta_X$ is bounded above and below by 1 and $10^{-5}$ respectively. Figure 4.13 contains realizations of the model with several extreme parameter values. Similarly to return mechanism I, the mean event heights of the mRNA $x$ and protein $X$ can be adjusted relative to the mean event height of total toxin (see Figure 4.13). The strength of the return mechanism (proportional to $\beta_X$) readily tunes the duration of toxin events for both of these situations as with return mechanism I. Figure 4.13 also contains several measurements of the event counts including the minimum number of events measured for return mechanism II (more than 1,300).

The toxic state probability is tunable across a similar range to that of return mechanism I ($\sim 0.0$ to $\sim 0.6$) using the considered return mechanism parameter values. The tunability is weakest at the extreme values of $\beta_X$ where the probability depends most weakly upon the transcription and translation rates ($\rho_X$ and $\alpha_X$ respectively). The dependence, and thus tunability, is maximized where $\beta_X$ is between the extreme values measured. The dependence of the mean event width upon the varied parameters again appears to be very similar to that of the toxic state probability (see Fig. 4.15). Mean event height again depends more gradually upon $\beta_X$, where peak tunability appears to be found at $\beta_X = 1$ (see Fig. 4.16). It is apparent that regulation of the affinity of either complexes or toxins promote similar tunability of the module. There are potentially several other ways of regulating toxin events via proteolysis, such as modulation of the maximal degradation rate $\mu$; however, modulation of enzymatic affinity is potentially simpler to achieve synthetically by introducing adaptor proteins in the presence of free toxin which help to degrade targets such as toxins or complexes.
Figure 4.13: Effect of return mechanism II on toxic events. Return mechanism II involves three parameters. \( \rho_X \) and \( \alpha_X \) are the transcription and translation rates of the additional species \( X \), and \( \beta_X \) regulates the affinity of the protease for the antitoxin bound to toxin in complexes. The top left plot shows that with fast production of the species \( X \) (\( \rho_X = 1.0 \) and \( \alpha_X = 1.0 \)) the proposed return mechanism can strongly reduce the duration of toxic events. The bottom left plot shows that with far lower \( \beta_X \), events are no longer stifled and the accumulation of \( X \) far outpaces that of the toxin. With far lower production parameter values, the top right and bottom right plots show that \( \beta_X \) can regulate the width of toxic events with relatively low accumulation of the additional species \( X \). The count of events used for statistics are also shown for several values of \( \beta_X \), where the minimum event count measured was 160. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are \( \sigma = 20 \), \( k_0 = 0.05 \), \( \nu = 1.25 \), \( \omega = 0.2 \), \( \beta_T = 5 \), \( \beta_g = 0.25 \), \( \mu = 100 \), \( \epsilon = 0.05 \), \( \gamma = 1 \), \( \alpha_t = 200 \), and \( k_C = 1000 \), while \( \rho_X \), \( \alpha_X \), and \( \beta_X \) are varied.
Figure 4.14: **Effect of return mechanism II on event probability.** A measurement of steady state toxic state probability as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
Figure 4.15: **Effect of return mechanism II on event width.** A measurement of the mean toxic event width as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
Figure 4.16: **Effect of return mechanism II on event height.** A measurement of the mean toxin count during toxic events as a function of the transcription rate $\rho_X$ and the translation rate $\alpha_X$ is shown for six values of $\beta_X$. Statistics were measured using single long trajectories of length 1,000,000 time units. Parameters are $\sigma = 20$, $k_0 = 0.05$, $\nu = 1.25$, $\omega = 0.2$, $\beta_T = 5$, $\beta_g = 0.25$, $\mu = 100$, $\epsilon = 0.05$, $\gamma = 1$, $\alpha_t = 200$, and $k_C = 1000$, while $\rho_X$, $\alpha_X$, and $\beta_X$ are varied.
4.5 Conclusions

In this chapter, we examined the role of proteolysis in promoting bacterial persistence through regulation of TA modules. In particular, we showed that two TA modules coupled through proteolytic competition have profoundly different toxic event statistics corresponding to persistence. Slowed cell growth rate concomitant with TA module borne persistence inherently couples modules, an effect which was examined as well as proteolytic coupling. With or without this source of module coupling, proteolytic competition strongly dilates the toxic events of each module and correlates the entry and exit of the toxic state between modules. This suggests that coordination of multiple TA modules through proteolytic competition could provide cells with regulatory mechanisms for their occupancy of the persistent state. These mechanisms could be the source of reliability which allows bacterial populations to achieve robustness in so many harsh and fluctuating environments. It is conceivable that drugs which interfere with proteolytic crosstalk might affect the ability of bacteria to regulate their persistence. For *M. tuberculosis*, which kills approximately 1.6 million people per year [30,31] and is known for its particular ability to maintain long and robust persistent states [95,110], this could be crucial for effective treatment.

We also presented two theoretical mechanisms where regulation of proteolytic specificity was shown to be capable of tuning the toxic events of a TA module. TA modules have not, as far as we know, been used in a synthetic context. For industrial applications, synthetic circuits will likely require robustness on par with natural bacterial populations. The ability to engineer robustness, or more specifically a persistent response, could be key to the development of industrial synthetic systems. In addition to synthetic applications, drugs could be developed which regulate proteolytic specificity directly, perhaps by inhibiting production of necessary chaperones for degradation (a TA module operon in *M. tuberculosis* has been shown to contain a third protein which functions as a chaperone for proteolytic degradation of the antitoxin [42]), or by regulating proteolysis of inspecific targets so as to prevent bottlenecks from forming which coordinate toxic state entry and exit between modules. It could be that aiding bacteria in their processing lowers their defense against antibiotics by shaping the statistics of persistence which depend upon environmental stress cues. While the modules considered in this chapter are identical and based somewhat specifically on the translational inhibition mechanism and complex stoichiometric ratio characteristic of the mazEF module in *E. coli*, proteolytic degradation of antitoxins is a common feature of many TA modules which differ in complex formation and deleterious toxic effects [66,68]. Moreover, organisms may possess many distinct modules which are similar to mazEF (9 distinct modules in *M. tuberculosis* are similar in function to mazEF in *E. coli* [95].) The most applicable conclusion to draw from the results presented here is that proteolytic regulation presents an attractive avenue in the pursuit of medicine which combats bacterial persistence.
4.6 Acknowledgements

Funding for this research was provided by the National Science Foundation under Grant No. MCB-1330180.
Chapter 5

Modular: A Simulation Framework in Python

5.1 Abstract

Computational methods have long been a pillar of modern science. Just as new experimental methods expand the boundaries of exploration, new algorithms and software are a constant byproduct of science which expand the range of tractable problems. Here we present Modular, a software package intended to function as a glue for computational techniques used in science, unifying methods under a common framework and workflow. While Modular is not limited to a particular variety of simulation, currently support extends primarily to chemical networks. By providing an extensible framework for analysis, Modular enables the rapid addition of support for other varieties of models as well as analytic routines. Freely available and with an emphasis on parallelization using computing clusters, Modular is a simulation package from which those seeking to perform large-scale simulation and analysis might benefit.

5.2 Introduction

Computational methods provide an indispensable perspective which serves many roles within science. Among the computational methods available, statistical analysis of ensembles of stochastic realizations of a model is relatively common [9, 18, 49, 50, 69–71, 78]. This is particularly relevant in fields such as biology, where stochasticity can play a fundamental role in the nature of system dynamics [9, 26, 63, 81, 104]. Simulation results are sometimes compared to experimental or analytic results in the interest of validating a particular model, or considered before significant experimental overhead is spent in the interest of streamlining the
Chapter 5. Simulation Software

process of engineering [53, 88]. Tools capable of performing simulations and analyzing their results are typically very specialized, serving only small niches of the scientific community. Some tools do not provide mechanisms to perform additional analysis beyond the extraction of several common statistics (e.g., mean and standard deviation).

*Modular* is a software package which fills a more abstract niche than many software packages, with the underlying goal of a unified pipeline that efficiently handles the many tasks which most such software packages depend upon for their utility. For example, it is common to aggregate simulation results of one or many realizations of a model given a distinct set of parameter values. Analysis usually consists of calculating statistics on a set of such results for each set of parameter values which is considered. This permits a measurement of statistics as a function of various parameters of a model (“parameter scan”) [9, 18, 49, 50, 69–71, 78]. None of the above process requires identification of how a realization of the model is achieved. *Modular* provides the capability of scanning parameters in such a way that the mechanism of model realization is interchangeable, and thus provides a piece of a pipeline common to most simulation work. The benefit of such an approach is that many projects may use the optimizations and advancements of others despite exploring very different models, provided such advancements are associated with a model independent routine as implemented within *Modular*. The range of such features extends well beyond scanning parameters (e.g., statistical analysis, plotting of results, and leveraging computing clusters).

*Modular* currently provides capable support for chemical network simulation. This is achieved via two additional Python packages which are included with *Modular* by default (*Gillespiem* and *Dstoolm*). These packages provide the necessary code to perform one realization of a model and act as plugins to the pipeline of *Modular*. *Gillespiem* provides an implementation of the Gillespie algorithm that is exact (simplified next reaction method [38]). *Dstoolm* provides simple ODE integration by using another Python package (“PyDSTool” [17]). *Modular* is capable of simulating models using either of these methods, where all processing downstream of the model realizations is essentially unaware of a difference. Thus, support for other simulation varieties may be added by implementing packages analogous to *Gillespiem* and *Dstoolm*, which provide the means of a single realization of other types of models. This is a considerably easier task than implementing everything needed to explore a model from scratch.

While this is the first work formally introducing *Modular*, it was used in the generation of results for several articles. In [9], support for arbitrary time dependent signals aided in the confirmation of results (published result arose from an earlier version of *Modular*). In [49, 78], the usage of a computing cluster as well as some basic statistical measurements was critical in generating large parameter scan results. All simulation results shown in previous chapters (with the exception of Figure 3.5) were generated using either previous or the current version of *Modular*. Older versions of *Modular* also provided support for the experimental methods used in [9, 49] (see chapter 6 for more information).
5.3 Major Pipeline Features

Below are descriptions of some of the major features of the Modular simulation pipeline. Specifically, the features below are strictly independent of any particular simulation method, and are thus available to any simulation plugin written for usage with Modular (e.g. Gillespiem and Dstoolm).

5.3.1 Mcfg File Parsing

Modular is run using configuration files, which are human readable text files following a specific syntax and ending with the file extension “.mcfg”. An mcfg file may contain three varieties of lines of text. Any line beginning with the “#” character is considered a comment and is ignored. Any line containing a phrase beginning with the “<” character and ending with the “>” character sets the active parsing function. Any other line is fed to the active parsing function resulting in some input to the simulation run. Almost anything about a simulation run can be specified within an mcfg file. Most information about a run can be left out of the mcfg file, allowing Modular to impose intelligent default values when possible. Simulation plugins such as Gillespiem provide additional parsing functions for input specific to models which are appropriate for the Gillespie algorithm (e.g. reactions and species). Example mcfg files are provided with Modular which demonstrate usage of core features of the package.

5.3.2 Extensible Measurements

Often statistical measurement of simulation results involves a mixture of generic statistical quantities and possibly nontrivial customized statistics [9,49,50]. Support for arbitrary measurement within the Modular simulation pipeline is achieved via implementation of subclasses of a base measurement class. Such a measurement may have as input the raw simulation results or the output of another measurement. This provides a flexible means to measure arbitrary quantities by chaining measurements together and most importantly an extensible framework for analysis.

Modular imposes at least one limitation on these measurements which permits an enormous optimization. Measurements which consume raw simulation data (one or more realizations of a model across one or more distinct sets of parameter values) may only access the data associated with a single set of parameter values at a time (for large scale simulations, having to maintain raw simulation data for more than a few sets of parameter values can be infeasible, or in the very least impractical and slow). Measurements of quantities as functions of model parameters are thus achieved through two measurements. The first measurement must act directly on one or more realizations of the model associated with one distinct set
of parameter values. The second measurement acts on the first measurement but has access to the results associated with each set of parameter values. Each measurement is permitted to perform its own output, allowing examination of the intermediate results of analysis.

Many commonly desired statistical measurements are currently supported and are immediately available to a new user. Among these, many users will find the measurement of means, medians, standard deviations, and correlations (Pearson correlation coefficient) to be useful. Measurements which reorganize other measurements based on the parameter space and extract subsets of the raw simulation data are also provided. Usage of measurements depends on the addition of specific lines to the relevant mcfg file, examples of which are found within the Modular repository. Examples of implementing additional measurement classes are also provided.

### 5.3.3 Computing Clusters

A common shortcoming of existing simulation software is a lack of support for computing clusters. Using the Python package MPI4Py [19], Modular may break up simulation work across many processes, which may or may not be bound to one computer. MPI4Py provides an implementation of the message passing interface (MPI) with bindings for the Python Programming Language. This permits flexibility in that Modular may utilize any number of processes on any number of computers (provided they are on the same network as is standard for computing clusters) with all of the runtime options afforded by MPI. By leveraging MPI, Modular works well with schedulers (e.g. PBS) as is often required for large clusters such as those found at many universities. This is in no way required, and Modular is capable of running on many machines without a scheduler, making it ideal for home use with multiple machines. As MPI does not require more than one machine, this implicitly permits leverage of multiple cores as are generally found in modern computers and as is more commonly available in simulation software packages.

### 5.3.4 Parameter Scanning

Some software packages provide a version of parameter scanning capability (e.g. COPASI [99]). As Modular is designed for large-scale simulations, there is no limitation on the dimensionality of a parameter space being explored. Furthermore, there is no limitation on the number of values which a particular parameter might take (aside from user specified limitations). The parameter space of a model being considered during a simulation run must be represented in the mcfg file. Typically this specification includes which model parameters are to be axes of the space, the bounds of each axis, the initial values to be used for each axis, the number of values to use to span the bounds of the axis, and the number of realizations required for each set of parameter values. The trajectory through the parameter space will contain each member of the Cartesian product of the values found along each axis.
5.3.5 Simulated Annealing

Parameter estimation is another capability sometimes found in simulation software packages (e.g. COPASI [99]). Modular is potentially distinct from other such packages in that it maintains the ability to leverage computing clusters in this context, allowing deep and rapid exploration of immense parameter spaces. At its core, the parameter estimation capability of Modular is a modified implementation of the well documented algorithm known as “simulated annealing” [25]. Parameter estimation also requires input data which is measured against the results associated with each attempted set of parameters. Such input data may not have the same resolution as the data generated during simulation (depends on the total simulation time and the number of desired output points). Modular will use interpolation to ensure that comparison of input data to simulation data is over the same domain with the same resolution. Furthermore, the user may be aware that some parts of the input data are more important to properly fit than other parts. For this circumstance, a weight function may be specified which biases the fit towards specific portions of the domain over which the fit is performed.

The basic simulated annealing algorithm used by Modular consists of a loop with only a few simple steps. Using the current parameter values and a parameter referred to as the “temperature”, a new set of parameter values is generated. The new set of parameters is a randomized perturbation of the current parameter values where the strength of the perturbation is proportional to the temperature. Simulation of the new set of parameters yields some data which is compared to the input data. This comparison must yield a single nonnegative value which indicates the discrepancy between the newly simulated data and the input data. If this discrepancy is lower for the new set of parameters, this new set of parameters becomes the current set of parameters. The temperature parameter is then lowered according to some monotonically decreasing nonnegative function (e.g. decaying exponential curve). Modular may use many processes to effectively try more sets of distinct parameters per unit of time, thus achieving far closer fits in far less overall time.

Modular is capable of treating the parameter space as discrete, where the values considered represent very different orders of magnitude. Perhaps the most effective routine available when the dimensionality of the space is high, the upper and lower bounds of the axes differ by many orders of magnitude, or the initial parameter values are orders of magnitude from the best values, is to iteratively perform this discrete fitting method, removing seemingly irrelevant portions of the space and repeating the search on the remaining space. Following several discretized fitting iterations with several continuous fitting iterations provides a mechanism for finding a decent parameter fit despite knowing effectively nothing about the correct parameter values even when a model has many unknown parameters. Figure 5.1 shows an example result of this routine for a simple enzymatic system where a single class of protease degrades a single class of substrate.
Chapter 5. Simulation Software

Figure 5.1: Simulated annealing result for simple enzymatic system. Using 50 processes on a laptop with four cores, the above fit was found in fewer than two minutes using a series of discrete and continuous fitting iterations. The simple network used has three reaction rates (binding rate, unbinding rate, and enzymatic processing rate) which were used as axes of the parameter space over which to perform parameter estimation. Each axis was bounded below by $10^{-7}$ and above by $10^6$ and set at an initial value of 1, emulating the situation where essentially nothing is known about the correct values of the parameters. The input data set used was a single realization using the parameter values 1, 0.01, and 800 for the binding, unbinding, and processing rates respectively. The parameters found to have the best fit were 0.1676, 0.0001, and 786.6.

5.3.6 Plotting Interface

Of the output formats supported by Modular, perhaps the most useful is the serialized format provided by the cPickle module of the Python standard library. Not only does this make the data easily available to any amateur Python programmer, it allows the user to immediately view the results using an interface provided by Modular. The interface consists of single window associated with each input data file (output data from simulation runs) which operate independently of one another. This window permits control of many features of the resulting plot such as the axis labels, line colors, and whether or not the legend is present. Perhaps most importantly, the interface provides a method to save the resulting
plot in several image formats, including PDF. As Modular was designed with the goal of supporting potentially high dimensional parameter scans, the plotting interface provides controls which allow the user to efficiently peruse through subspaces of the parameter space which was measured. For instance, if a parameter space has five axes, and the desired plot is a heat map of some measurement as a function of two of those axes, the remaining three axes must be fixed at one value, designating a specific subspace for which to show a plot. A more detailed description of this interface is found with Figure 5.2.

![Figure 5.2: Plotting interface of Modular.](image)

The plotting interface of Modular consists of three major regions outlined in the above with three distinct colors. The green box outlines the resulting plot based on selections from the other two boxes. A toolbar below the plot provides the ability to pan, zoom, and save the image. The blue box contains various options for configuring the plot. Many of the controls are not visible in the above image, but this box contains controls for the axis labels, bounds, whether or not they are in log-space, which lines are visible, their widths, colors, symbols, and labels, and more. The red box contains a tree that associates a set of trajectories (only one trajectory in the provided image) with each distinct set of parameters which was considered in the run. For measurements which act on the entire parameter space, there is simply a set of trajectories unassociated with any particular position in parameter space.

### 5.4 Usage of Modular

Modular is currently available under the MIT license at http://github.com/ctogle/modular. This repository includes all necessary source code, documentation, demo code, and test code. While Modular will likely function on Mac OS X or Windows, the target operating system
is Linux, as this is most commonly found on computing clusters. The repository contains an executable Python script which performs installation without requiring root privileges. This script does not install the various Python packages on which Modular depends. Instead, a user is expected to maintain their own distribution of Python, as Modular should not dictate the underlying Python packages available to a user, nor should it require its own custom distribution. We recommend new users begin with the standard Anaconda distribution of Python 2\footnote{72}. In addition to this distribution, the following packages (all freely available) are required: appdirs\footnote{75}, MPI4Py\footnote{19}, and PySide\footnote{102} (strictly for the plotting interface). Support for Python 3 is currently absent but may be added in the future.

### 5.4.1 Typical Workflow

Modular is intended to provide a clean workflow for either local or remote usage (e.g. SSH). A short bash script ("mrun.sh") is provided with Modular which serves as the primary entry point for most workflows. This executable provides an easy mechanism to perform work using potentially many processes on one or more computers with or without the use of a scheduler. Everything that defines a simulation run and its analysis is provided in either the mcfg file or as a command line argument to this executable. The typical workflow for most users will consist of only several steps. First the mcfg file must be written. Using the mrun.sh script and this mcfg file, simulations and analysis are performed resulting in a set of output data files as specified in the mcfg file. If the work was performed on a remote machine, it is likely best to transfer the data files to a local machine which also has Modular available. Lastly, by providing a slightly different set of arguments to the mrun.sh script, these data files are opened using the plotting interface where the user may render figures with the appropriate axis labels, legend, line styles, etc. The interface provides a routine to save the final figure in the desired image format. More detailed information about this process is provided in the form of tutorials which are distributed via the Modular repository.

### 5.4.2 Python API

Fundamental to the design of Modular is the separation of interface from core routines which perform the actual simulations and analysis. This decision enables the end user to pick and choose which components of the Modular pipeline are actually used. At the center of the pipeline is an ensemble class which provides a capable API accessible from Python. More detailed information about usage of this class is provided in the form of tutorials which are distributed via the Modular repository. Below is a minimal Python script which demonstrates parsing an mcfg file, performing simulations and analysis, and outputting the results for viewing with the plotting interface.

```bash
#!/usr/bin/env python
```
import modular4.ensemble as me

if __name__ == '__main__':
    mcfg = './path/to/file.mcfg'
    mod = 'gillespiem'
    r = me.ensemble(module = mod).parse_mcfg(mcfg).run()
    for o in r:

To minimize the required code to run an mcfg file, most of how the simulations are performed (e.g. the desired parameter space, analytic routines, and output formats) is described in the mcfg file. Below is an example mcfg file (compatible with Gillespiem) demonstrating measurement of the phenomenon of correlation resonance for a simple proteolytic network similar to those in [18,71,78].

<end>
  time limit : 1000

<capture>
  increment : 10 : time

<variables>
  mu : 10.0
  K : 0.00001
  lambda1 : 5.0
  lambda2 : 5.0
  gamma : 0.01

<functions>
  g : mu/(K + x1 + x2)

<reactions>
  null lambda1 -> 1 x1 : formation of x1
  null lambda2 -> 1 x2 : formation of x2
  1 x1 gamma -> null : dilution of x1
  1 x2 gamma -> null : dilution of x2
  1 x1 g -> null : degradation of x1
  1 x2 g -> null : degradation of x2

<species>
  x2 : 10
  x1 : 10

<targets>
Chapter 5. Simulation Software

Modular also contains simple convenience classes for creating multipanel figures using output data as generated by the above mcfg file and the Python plotting package Matplotlib [52]. Below is an example Python script which uses these classes to create and display a figure showing correlation resonance (also below).

```python
#!/usr/bin/env python

import modular4.mplt as mt
import matplotlib.pyplot as plt

def heat(p, x, y, z, x1, y1, pl, l1, l2):
    kws = {
        'xmin':1,'xmax':10,'ymin':1,
        'ymax':10,'zmin':0,'zmax':10,
        'xlab':x1,'ylab':y1,'plab':pl,
    }
    msub = mp.subplot(p,**kws)
    msub.add_heat(x, y, z)
    msub = mp.subplot(p,**kws)
    msub.add_line([[l1,l2],[l2,l1]],
                  name = '',width = 5,style = '--')

if __name__ == '__main__':
    mp = mt.mplot(pipe = [0,4,8,12,16,20])
```
mp.open_data('./demodata/bypsp_output.pkl')
x, y, z = 'lambda1', 'lambda2', 'x1, x2 - correlation'
heat('231', x, y, z, x, y, z+ (mu = 0)', 0, 0)
heat('232', x, y, z, x, y, z+ (mu = 4)', 1, 3)
heat('233', x, y, z, x, y, z+ (mu = 8)', 1, 7)
heat('234', x, y, z, x, y, z+ (mu = 12)', 1, 11)
heat('235', x, y, z, x, y, z+ (mu = 16)', 1, 15)
heat('236', x, y, z, x, y, z+ (mu = 20)', 1, 19)

mp.render()
plt.show()

Figure 5.3: **Correlation resonance for a simple proteolytic network.** Correlation resonance is shown for several parameter spaces explored by the three dimensional parameter scan described in the above example *mcfg* file. This measurement required over 8.2 million realizations of the Gillespie algorithm and was performed on two laptops (6 processing cores total using MPI) in fewer than 100 minutes.
5.4.3 Extending *Modular*

Extensibility is a key consideration in the design of *Modular*. The addition of either measurement classes (analytic routines) or simulators (analogous to *Gillespie* and *Dstool* ) is intended to be easy; however, this process does require some knowledge of Python programming as both measurement classes and simulators are written in Python (perhaps appealing to libraries written in C using the Python package Cython [6]). More detailed information about extending *Modular* with measurements and simulators is provided in the form of tutorials which are distributed via the *Modular* repository.

5.5 Current Limitations of Modular

As mentioned, two varieties of simulation are currently available for models of chemical networks (*Gillespie* and *Dstool*). New simulation varieties can be added by implementing more analogous packages; however, such packages must be usable from the Python Programming Language. Fortunately, Python is a particularly extensible language specifically with many existing tutorials on making code written in C available to Python by using the Python API. This may still seem like a stark limitation if not for the Python package Cython [6]. Cython is package which eases the extension of Python by automatically generating C code from Python code which is then easily used to extend Python. Files which contain Python code to be used in this manner may contain additional information which helps Cython generate efficient code that competes in speed with C. Moreover, such files may import C libraries directly, allowing the extension of Python with C code that was never intended to be used with Python. This provides both a mechanism to optimize troublesome bottlenecks of Python code as well as bind arbitrary C libraries to Python. This particular inconvenience is best left to a relatively experienced Python programmer, but only requires a single solution from which anyone else could benefit. As both *Gillespie* and *Dstool* function with *Modular*, this problem is effectively already solved for Gillespie simulations and coupled first-order ordinary differential equations.

As the data used for parameter estimation may come from another software package or perhaps from experiment, it is very difficult to create a single routine which converts arbitrary potential input to the correct format. This process must instead be left to the end user, who is responsible for understanding the nature of their data. Some auxillary scripts are provided which demonstrate the conversion of some data formats to the required format for input data to *Modular*. This requires far less understanding of Python than the previously mentioned limitation, but again only a single solution must be generated for others to benefit.

As a process must have sufficient memory to contain the raw simulation data of at least one set of parameter values, there is a hardware limitation on the number of realizations which can be performed for a set of parameters. *Modular* allows specification of the output
resolution, which when low permits the aggregation of more than a sufficient number of realizations to measure useful statistics. For example, the statistical measurements presented in chapter 3 were made across ensembles of 20,000 realizations, which is far more than sufficient for useful statistics.

5.6 Conclusions

Here we presented the software package **Modular** for Python. By compartmentalizing various features of a simulation pipeline, **Modular** provides a flexible and extensible framework for simulation and analysis. As the use of computing clusters is central to the design of **Modular**, and because there is no limitation on the dimensionality of a parameter space, it provides a scalable means to explore potentially very large parameter spaces. By integrating measurement into the process of generating raw simulation data, **Modular** achieves an extensible analytic pipeline. Most importantly, by isolating the role of a simulator to the generation of a single realization of a model for a single set of parameters, **Modular** provides a framework from which many could potentially benefit. Potential new users who seek to explore chemical networks need not have any knowledge of programming as simulation of such models is currently supported. Additional simulation types may be supported by following tutorials found in the **Modular** repository. For most potential analysis, only minimal Python knowledge is necessary, and tutorials are provided which demonstrate the process of adding analytic routines (measurement classes). Many common statistical measurements are already supported and may immediately be used with any variety of simulation. **Modular** thus targets as a user those with and without programming knowledge, possibly seeking to perform any variety of simulation. Lastly, **Modular** is freely available, and has been produced in the spirit of collaboration in the hope that more capability can be added in the future from which many potential users might benefit.

5.7 Acknowledgements

Funding for this research was provided by the National Science Foundation under Grant No. MCB-1330180. We would also like to thank Matthew Via and Justin Bangerter for indispensable technical advice on the development of software as well as the Python community at large for having provided the capable set of packages on which **Modular** depends.
Chapter 6

Software Support for Microfluidic Experiments

Previous versions of the software presented in Chapter 5 were used to construct two additional applications used during microfluidic experiments [9, 49]. The ability to provide fast and accurate signaling to the cells within the chip is accomplished by positioning a syringe relative to the chip such that gravity provides sufficient pressure to force the contents of the syringe to flow into the chip. The position of the syringe is modulated using a stepper motor which is interfaced using the Python bindings provided by the Phidgets library. These bindings permit many controls for how the stepper motor operates, including maximum angular velocity and acceleration, which were necessary for accurate signaling.

One of the additional applications is referred to as the “receiver.” The receiver is responsible for the stepper motor input and provides the interface for controlling how the motor moves. This includes the velocity and acceleration, but more importantly it includes the bounds for the position of the motor which is characteristic of the experiment and the required pressure to provide the proper concentration of the contents of the syringe (contents are mixed within the chip such that the motor position regulates the concentration). The receiver works by establishing a list of instructions (positions to move to) and associated times to execute those instructions. There are several ways to provide an instruction, the most intuitive of which are several buttons in the interface for moving the motor to either extreme position. The more flexible approach which was actually used during experiments involved the parsing of a time-dependent signal from a text file. Each line of the file provided a time and a position for the motor to move to at that time.

Execution of instructions takes time which depends on how far the motor is from the next position and how quickly it is permitted to move there. It is best to simply determine the time it takes to move from one extreme position to the other, and for the signal to never require two instructions which are not separated by that amount of time or more. The receiver handles this situation by simply ignoring an upcoming instruction if it has not
finished executing the current instruction. The completion of an instruction requires that the current motor position is within a specified tolerance (units specific to motor) of the target position.

The second application is referred to as the “transceiver.” The transceiver provides additional buttons for moving the motor immediately to its extreme positions as well as controls to select and impose a time-dependent signal from a text file. The transceiver communicates with the receiver using UDP messages that contain a time to execute an instruction and the target position of that instruction. Signals which are contained in such text files are parsed into a set of instructions which are passed along to the receiver. The transceiver may or may not be run on the same machine as the receiver, which for our experimental setup was run on a computer responsible for taking fluorescent images. The primary reason to allow this separation was to permit a possibly computationally intensive operation to determine the proper instructions at run time. More specifically, the goal was use image analysis on the resulting data from microscopy images to direct the motor using a feedback loop. One potential application for such a capability is to create a cell sorting device, which places cells in one place or another based on dynamics that can be measured during the experiment (e.g. fluorescent light intensity).

While these two applications were sufficient to conduct microfluidic experiments within our lab, there is much room for improvement. Firstly, the dependence of both applications upon older versions of Modular will eventually become a problem as the older versions of Modular will not remain maintainable. This dependence arises from code used to create the interface which is no longer used in current versions of Modular. While it is not a trivial undertaking to remove this dependence, it is recommended for those who wish to conduct more elaborate experiments than those found in [9, 49]. The current receiver implementation is limited to control of one motor, whereas the freedom to control more motors would greatly expand the range of possible experiments. There is significant value in separating motor control from motor instruction determination which should permit complicated experimental methods that require feedback. These applications should serve as a useful starting point for constructing better implementations of analogous applications required for some microfluidic experiments.
Chapter 7

Conclusions

Cell biology is a rapidly evolving field that requires the advancement of experimental and computational methods. Synthetic biology provides many experimental methods as well as a platform for the development of technology from which many could benefit. Despite the historical intersection of these fields at *Escherichia coli*, much has been learned about more complicated organisms composed of drastically different cells. Enzymatic processing, a relevant feature for all cellular systems, was of foremost interest in the previous chapters for the necessity of its understanding. While these chapters focused specifically on proteolysis, the implications of their conclusions have potentially far reaching impact on the field. Bottlenecks owed to enzymatic limitations not only induce drastic differences in molecular dynamics but also aid cells in promoting responses to the fluctuations in their environments. Moreover, queueing theory continues to provide a natural vehicle for the exploration of bottlenecks in biological chemical networks.

In the context of synthetic biology, coupling afforded by proteolytic crosstalk may be used to couple signals of otherwise noninteracting gene networks. By modulating secondary components such as adaptors, one could conceivably make this coupling conditional upon some chemical additive. In the context of persistence, the coupling of toxin-antitoxin modules is of interest as it is well known that many modules could potentially be naturally coupled through proteolysis. The understanding of such dynamics may be useful to those developing treatments which target toxin-antitoxin modules, perhaps breaking the coordination achieved via proteolytic competition.

For the computational explorations shown in previous chapters, new software was developed which remains freely available online. Specifically, investigation of chemical networks using computing clusters has proven effective, while *Modular* constitutes a capable and extensible tool to this end. While this software was used in the construction of additional software used for microfluidic experiments, these additional pieces of software will not fulfill all experimental setups. Nonetheless, it is hoped that *Modular* and these additional pieces of software may aid others wishing to simulate chemical networks or perform microfluidic techniques.
Bibliography


Chapter 7. Conclusions


Chapter 7. Conclusions


Chapter 7. Conclusions


[99] COPASI-A COmplex PAthway SImulator. Hoops.


Chapter 7. Conclusions


