BUILDING AND UNDERSTANDING: REPRESSILATORS AND SYNCHRONY

THREE CELLS IN A BUBBLE BATH

Brenton Munson Clara Posner Jonathan Pekar

October 23, 2016

UNIVERSITY OF CALIFORNIA SAN DIEGO
DEPARTMENT OF BIOENGINNERING
BENG 221 PROJECT

CONTENTS

1 Introduction 1
1.1 Problem Statement . . . 2
2 Methods 2
2.1 Simplifying and Parameterization . . . 2
2.2 Numerical Method . . . 5
3 Results and Discussion 6
References 14
4 Appendix 16
4.1 Code . . . 16
4.1.1 Single Cell Repressilator Model . . . 16
4.1.2 Repressilator Model with Drug Sensing . . . 23

LIST OF FIGURES

Figure 1 Repressilator System . . . 2
Figure 2 Repressilator with Drug Sensing . . . 4
Figure 3 Multicell Environment . . . 5
Figure 4 Starting a Repressilator . . . 7
Figure 5 Repressilator at Steady State . . . 8
Figure 6 Model Sensitivity to Transcription Rate . . . 8
Figure 7 Model Sensitivity to Translation Rate . . . 9
Figure 8 Model Sensitivity to Protein Degradation Rates . . . 10
Figure 9 Model Sensitivity to Hill Parameters . . . 11
Figure 10 Controlling a Repressilator with an External Stimulus . . . 11
Figure 11 Cell Synchronization . . . 12
Figure 12 Cell Synchronization . . . 12

1 INTRODUCTION

The demands placed upon modern medicine are rapidly driving scientists and engineers to investigate the mechanisms of disease and biological function to ever-increasing complexity. Much of this frontier lies in modeling not only the outcome of individual genetic perturbations but also how the networks of genetic interactions coordinate cellular- and tissue-level phenotypes [1]. While progress has been made in describing living systems, these networks often become too complex and cumbersome to interrogate with the current tools of bioinformatics [2, 3]. As such, the problem is often broken down into more discrete subunits, with which they can be modeled, perturbed, and hopefully understood on a more functional level [4]. In a classical engineering sense, these broken down sub-networks can be thought of as unit operations, and, once characterized, can be pieced together into higher order biological systems.

Synthetic biology is an emerging field at the intersection of engineering and molecular and cellular biology, where it attempts to create and model both novel and natural systems. There are two approaches within synthetic biology to tackle such problems: top-down and bottom-up [5]. The former attempts to take a living system, such as yeast, and insert designed biological units, often in the form of DNA plasmids. By leveraging the existing cellular infrastructure of a host organism, scientists can rapidly develop novel systems without needing to create an entire de novo system. However, this hijacking is not without consequences. While the systems of synthetic biology are currently relatively small, they can still impart stress on the host organism and change functions in unforeseen ways. Therefore, synthetic biology constructs (as well as all biological constructs) must be understood in the context of signaling and genetic regulatory networks in order to better understand their roles and consequences.

The bottom-up approach to synthetic biology can avoid much of the complications arising from using a host organism by taking non-living components and initializing the system ex-vivo [6]. However, designing and constructing such systems is largely outside our current technological purview. While a bottom-up approach will lend itself well to a more controlled environment for understanding synthetic constructs, current reports and models mostly leverage a top-down approach. However, when modeling regulatory networks in living systems, it is still useful to start from the fundamentals.

The central dogma of molecular biology describes how DNA is encoded variably into RNA, which is then translated into protein. The produced proteins are the machines of the cell, providing a vast array of functions and forms. Controlling this process is one of the fundamental roles of cellular function and is carried out by protein-based transcription factors. The first synthetic gene circuits implemented in biological hosts were the toggle switch and the repressilator. They have been used to study the dynamic behavior of transcription and translation, as well as to begin to develop theories and predictive capacities regarding synthetic networks [7, 8].

The toggle switch consists of a bi-repressive system of two genes, where the protein product of each gene acts as a repressor of the other gene, while also acting as a self-promoter. The

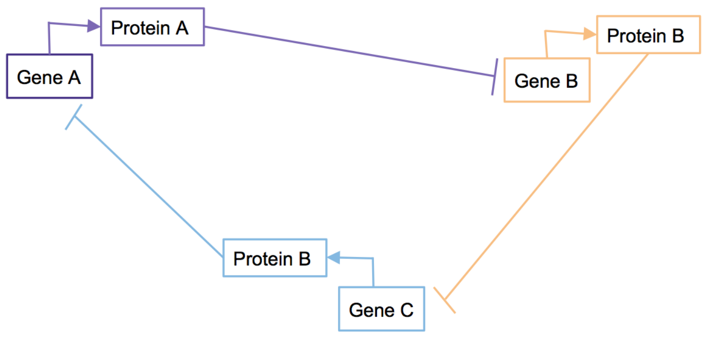

Figure 1: Repressilator System

system exhibits a switch-like behavior: one gene is constitutively expressed, while the other remains untranscribed. Bifurcation between bistable and monostable systems arises from varying repressor strengths and cooperativity. The repressilator takes the toggle switch one step further by introducing an additional repressing protein to the system, giving rise to oscillatory behavior. Here we investigate a mathematical model of the repressilator to better understand the dynamics and sensitivities of the system, model parameters, and propose a control mechanism by which oscillatory behavior can be temporally controlled.

1.1 Problem Statement

Genetic maps are often constructed without a mathematical framework that would help predict function and consequences when changes occur. Here, we look at the relationship among three genes that repress each other and model their relationship mathematically. We use numerical methods to elucidate the dynamic relationships arising from their map structure. We then use these methods to look at how a drug might induce synchrony among cells.

2 METHODS

2.1 Simplifying and Parameterization

We describe the repressilator system by a series of ordinary, nonlinear differential equations (equations 1 to 3). The main processes that change the concentration of the mRNA for each gene are degradation and repressible transcription. We assume the degradation is the product of the an mRNA degradation rate constant (k_{dm}) and the instantaneous mRNA concentration. The production of mRNA is the transcription rate α (currently assumed to be equal for all three genes), modulated by the concentration of the repressor. The Hill coefficient (n) accounts for the cooperativity of a protein in its repression of mRNA transcription.

$$\frac{da}{dt} = -k_{dm}a + \frac{\alpha}{1+C^n} \tag{1}$$
$$\frac{db}{dt} = -k_{dm}b + \frac{\alpha}{1+A^n} \tag{2}$$
$$\frac{dc}{dt} = -k_{dm}c + \frac{\alpha}{1+B^n} \tag{3}$$

The protein is created by translation, quantified by the product of the translation rate β and the mRNA concentration. We also assume the degradation is the product of the protein degradation rate constant (k_{dp}) and the instantaneous protein concentration.

$$\frac{dA}{dt} = \beta A - k_{dp}A \tag{4}$$
$$\frac{dB}{dt} = \beta B - k_{dp}B \tag{5}$$
$$\frac{dC}{dt} = \beta C - k_{dp}C \tag{6}$$

For our single cell simulation, we aim to create a repressilator system that will reach an oscillatory steady state involving its three genetic elements. To achieve this, we set the initial concentrations of all mRNA and protein to zero, except for the mRNA from gene A, which we set to 1M. We set the following parameter values: transcription rate (α) to 100, the Hill coefficient (n) to 2, the degradation constants of both mRNA and protein (k_{dm} and k_{dp}) to 1, and the translation rate (β) to 1.

To examine the effect of the different parameters on the system and its oscillatory nature, we vary the transcription rate (α), translation rate (β), and the Hill coefficient (n).

We can control the oscillatory nature of the repressilator system in one cell by introducing a drug D that induces the transcription of gene A at rate k. For this model, we assume the drug diffuses into the cell and the diffusion out of the cell is negligible. We assume that the cell is permeable to the drug inducer, so that an injection of concentrated drug into the extracellular media causes an equal concentration spike of drug concentration inside the cell.

The drug addition only modifies the rate of change of mRNA concentration for gene A; the previous mRNA and protein equations remain the same (Fig. 2). The drug is also assumed to degrade at a rate of k_{dd}, so that it is not continually activating gene A.

$$\frac{dA}{dt} = -k_{dm}a + \frac{\alpha}{1+C^n} + kD \tag{7}$$
$$\frac{dD}{dt} = -k_{dd}D \tag{8}$$

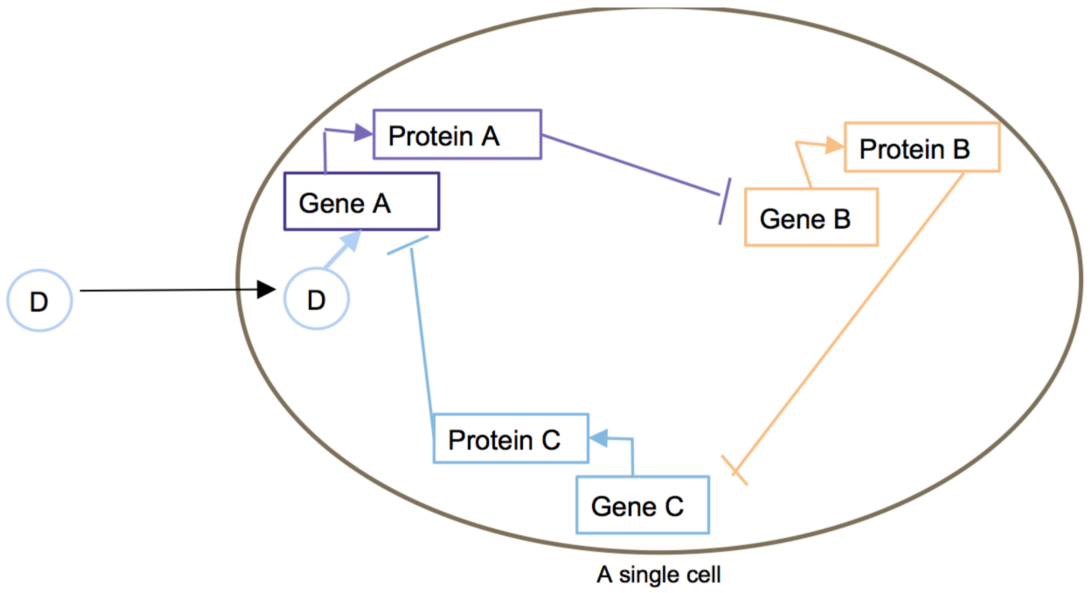

Figure 2: Diagram of the Repressilator System in a Single Cell with a Drug Diffusing into the Cell and Inducing Gene A

Our next modification to the system is to model three different cells all containing the repressilator system, but each cell is initialized with a non-zero mRNA concentration of a different gene (Fig. 3). We model a large spike of the drug inducer to the extracellular media surrounding all three cells by assuming that each cell uptakes the same amount of drug. This is to examine whether we can use an external agent to synchronize a group of asynchronous cells.

The differential equations used to model each cell in the three-cell system (shown in equations 9 to 15) are the same as the equations for the single cell with the added drug. The "i" denotes the identity of the cell as labeled by a number.

$$\frac{dA_i}{dt} = -k_{dm}a_i + \frac{\alpha}{1+C_i^n} + kD \tag{9}$$

$$\frac{db_i}{dt} = -k_{dm}b_i + \frac{\alpha}{1+A_i^n} \tag{10}$$

$$\frac{dc_i}{dt} = -k_{dm}c_i + \frac{\alpha}{1+B_i^n} \tag{11}$$

$$\frac{dA_i}{dt} = \beta A_i - k_{dp}A_i \tag{12}$$

$$\frac{dB_i}{dt} = \beta B_i - k_{dp}B_i \tag{13}$$

$$\frac{dC_i}{dt} = \beta C_i - k_{dp}C_i \tag{14}$$

$$\frac{dD}{dt} = -k_{dd}D \tag{15}$$

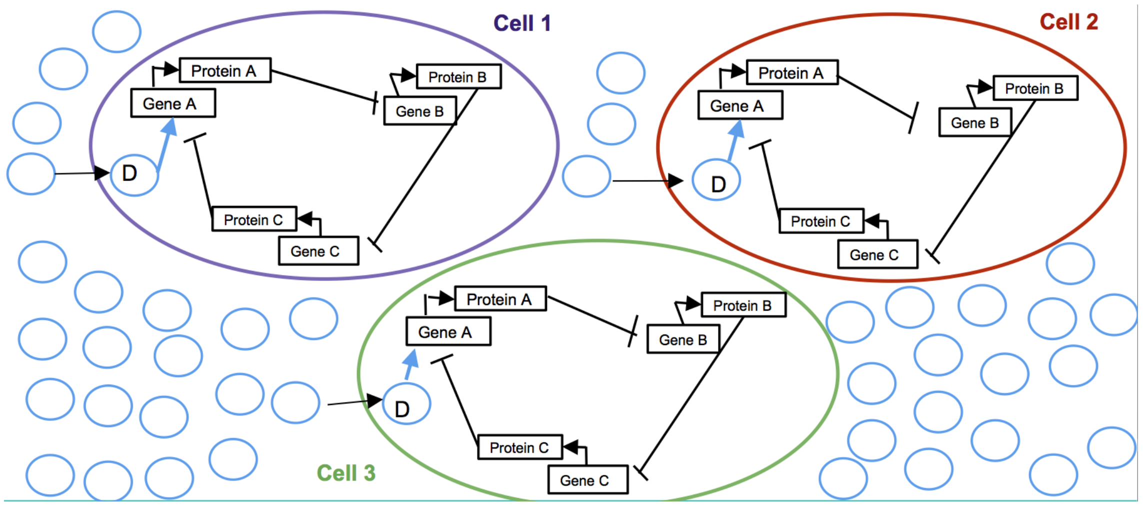

Figure 3: Diagram of the Three Cells containing the Repressilator with a Drug Diffusing into the Cell and Inducing Gene A.

Initially, all protein levels are set to zero. Cell 1 starts with a 1M mRNA for gene A, cell 2 starts with a 1M mRNA for gene B, and cell 3 starts with a 1M mRNA for gene C. The cells are allowed to reach steady-state oscillations before and after the drug is spiked in. The oscillations in each cell's protein A concentration are plotted together and compared to determine whether or not the spike of the drug inducer causes the cells' gene expression to oscillate in synchrony.

2.2 Numerical Method

Numerical integration of the six coupled ODEs were performed in the R statistical framework, using the ode function from the package deSolve with the Runge-Kutta 4 (RK4) parameter method [9]. The RK4 method is a specific implementation of the generalized Runge-Kutta family of algorithms. These algorithms all share the commonality of estimating a function value as the sum of a previous known value and the product of a characteristic slope and step size.

$$y_{n+1} = y_n + \phi h \tag{16}$$

The characteristic slope, ϕ, is approximated differently for specific implementations of the Runge-Kutta algorithm. In the RK4 method, this slope is taken as the weighted average of the functions slope at 4 points along a step size interval.

$$y_{n+1} = y_n + (a_1 k_1 + a_2 k_2 + a_3 k_3 + a_4 k_4)\, h$$

Here successive slope estimates, k_i, occur at function values derived from the previous slope estimate, k_{i-1} for $i > 1$. In the x domain, time in the repressilator model, the k slope estimates are taken at the beginning of the interval x_n for k_1, at the interval midpoint, $x_n + h/2$ for k_2 and k_3, and at the end of the interval $x_n + h$ for k_4. As such the k values are computed as:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1 h\right) \\ k_3 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2 h\right) \\ k_4 &= hf(x_n + h, y_n + k_3 h) \end{aligned} \tag{17}$$

The weights of are derived from a Taylor series expansion of y about y_i and while not included here, result in n equations and $n + 1$ unknowns. Where n is the number order of the Runge-Kutta method; in RK4 the order is four. Consequently, the an assumption about one of the weights is required in all of the Runge-Kutta algorithms [10]. As such, each time point in can be approximated as:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{18}$$

The RK4 method provides a good balance between accuracy and computational weight, with an error proportional to the step size to the fifth power. As such, it presents a large improvement over finite differences and Euler's method, where the error scales as the step size, h, and the cube of the step size, . While the four parameter method does require additional terms, it is the point at which improvements in accuracy no longer scale linearly with additional parameters [11].

3 RESULTS AND DISCUSSION

To understand the dyanmics of the system we wanted to create, we first had to model the basic version: one cell, three repressive genes, and no drug. With an initial concentration of 1 of $mRNA_A$, the concentration of $mRNA_A$ and $protein_A$ experience the largest initial increase, with the products of gene C experiencing the lowest increase, and B in the middle (Fig. 4). Then, within 40 seconds, the the products of all three genes are experiencing their maximum amplitudes and oscillations. Oscillatory behavior is readily visible; the genes (and more specifically, the proteins) cycle in their degrees of repressing one another, leading to cycles in which protein and mRNA species are the greatest at a given moment in time (Fig. 5a). Despite a transcription rate of 100, the amplitude of the mRNA is roughly 80-85; this is indicative of the mRNA simultaneously degrading while being transcribed. Similarly, despite a 1:1 rate of translation, the protein amplitude is slightly lower than that of the mRNA, showcasing that the mRNA degrades slightly before translation even begins, as well as that protein is also continually degrading. The plot in Fig. 5b showcases the dynamics of the protein and mRNA concentration for a given gene. As the protein and mRNA concentrations each slowly increase, they eventually begin to concomitantely enter a an oscillatory cycle; the concentrations first increase (moving upward and to the right) and then decrease (moving back down and to the left), slowly increasing their maximum concentrations (as seen by the growing loops), until they finally enter the largest, outer loop and repeat the same pattern of growth and decay.

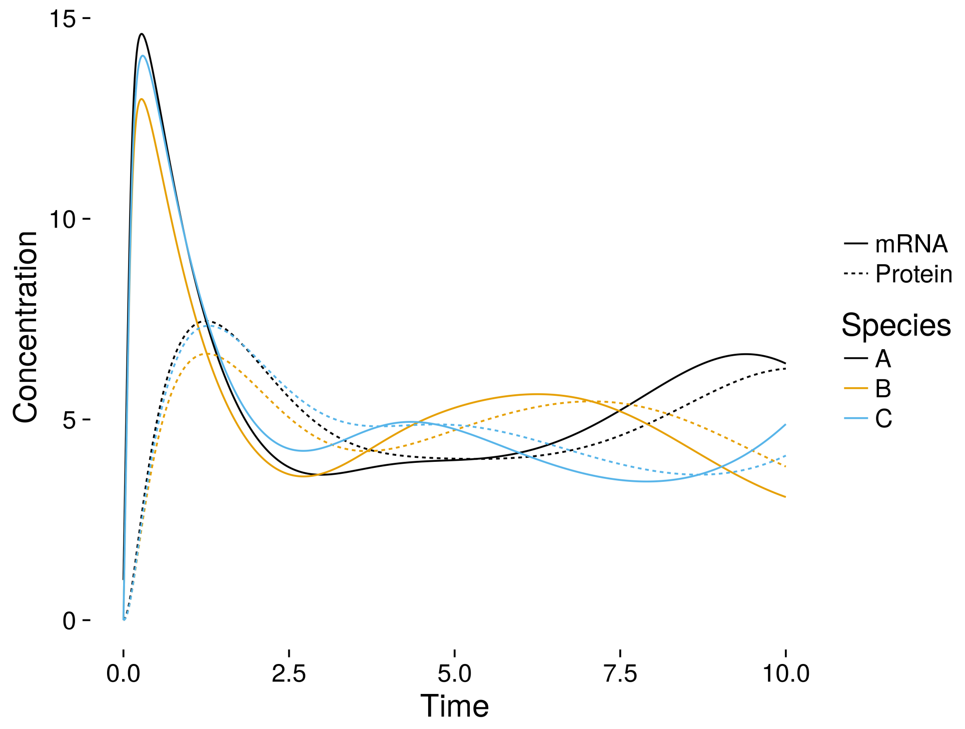

Figure 4: Initial transient dynamics of a repressilator system initialized with only one mRNA species present.

We first decided to see the dynamic effects from manipulating transcription rates. We found that oscillatory behavior is maintained when the rate is 10 and above, but appears to disappear at $\alpha = 1$ (Fig. 6). The low rate of generation of mRNA leads to a rate of translation that is so low, that we suspect the proteins are hardly repressing the genes; this low rate of repression most likely is the cause of lack of oscillation. The mRNA and protein quickly reach steady state. When we zoomed in on the steady state curve (not shown), there is a minimal level of oscillation, but the curves are converging.

We initially set β equal to 1, indicating that each molecule of mRNA translates one protein; a translation rate of 1:1. Here, we see rates of protein generation that are slightly below the rates of mRNA generation; this is due to the slight degrading of mRNA before it actually translates the protein (Fig. 7). When β is dropped to 0.1, ten molecules of protein are required per protein. Therefore, on the chart the oscillations and concentrations of mRNA are much greater than that of protein; a lot of mRNA will be required to generate a small amount of protein. Conversely, when β is increased to 5 or 10, much less mRNA is needed to generate protein, and the graphs showcase much greater oscillations and concentrations of protein compared to mRNA. Similarly, the rate of

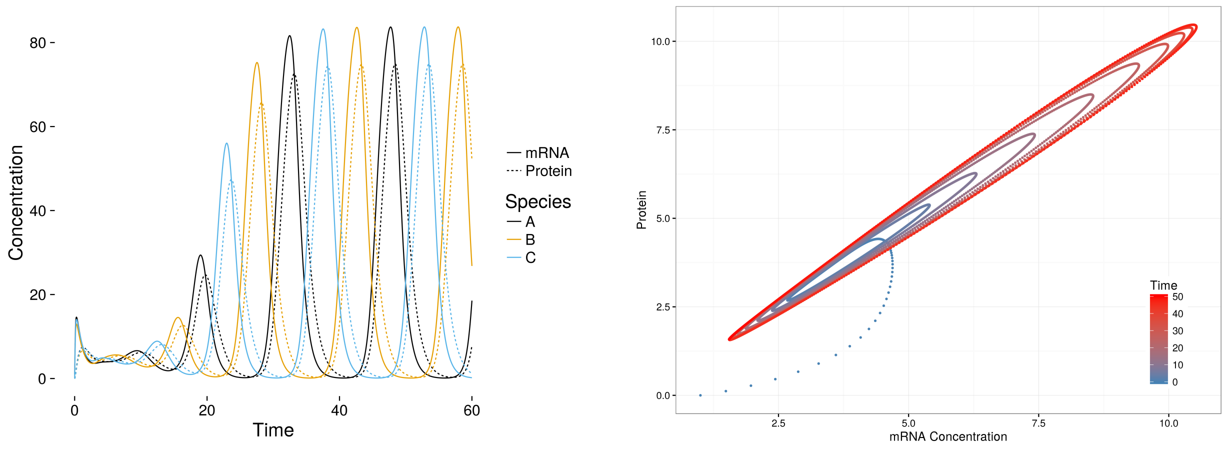

Figure 5: (a). Long term steady state mRNA and protein oscillations of a single cell repressilator system initialized with only one mRNA species present. (b). Oscillatory behavior of a single species mRNA and protein shows linear dependence with increasing magnitude of oscillations until steady state is reached.

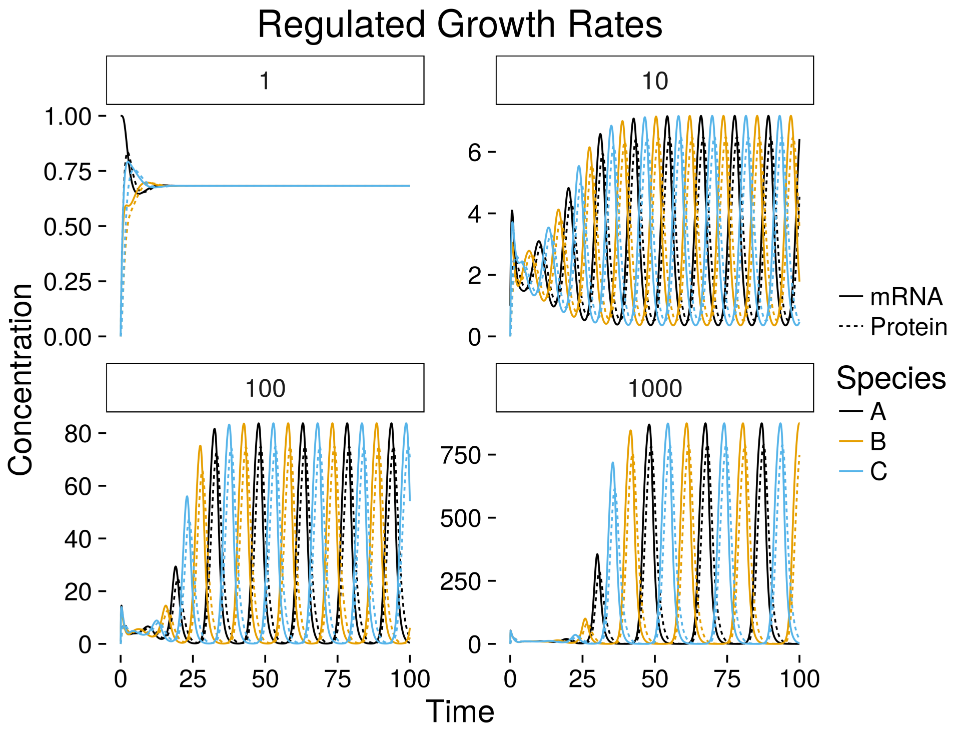

Figure 6: Steady state behavior of a single cell repressilator for varying the rates of transcription, α.

protein degradation plays a role in steady state oscillatory. For proteins which are quick to degrade after translation, oscillations degrade into equality for each of the three genes (Fig. 8).

Cooperativity is when the binding of a ligand is enhanced due to other ligands being present on the molecule, and this is described by the Hill equation [12]. The Hill coefficient is the variable used to quantify this effect, and we used this method to describe the cooperative repression of

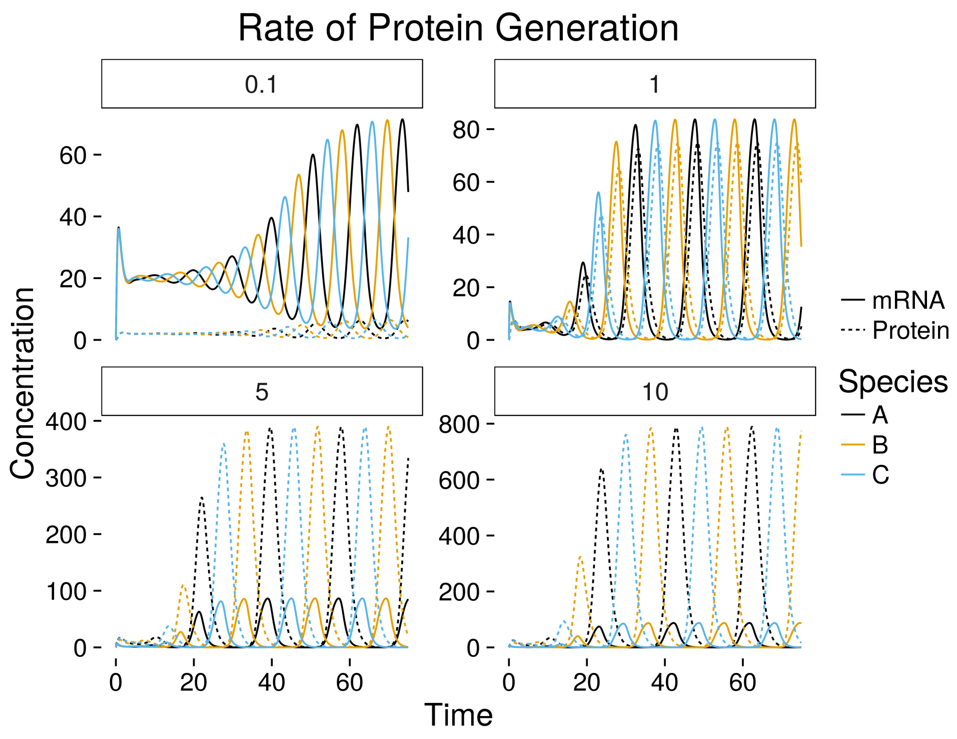

Figure 7: Steady state behavior of a single cell repressilator for varying the rates of translation, k_{gp}.

a gene by a protein. With a Hill coefficient of two, we see a small level of cooperativity, which leads to fairly standard oscillatory behavior. When we increase the Hill coefficient to 10, the oscillations become much steeper, with sudden drops in the oscillation compared to what is seen with a Hill coefficient of 2 (Fig. 9). The rate of cooperativity, and therefore the rate of repression, is so much higher with this greater Hill coefficient, that the moment a protein begins to repress a gene, the gene is extremely repressed and the mRNA and protein concentrations of the given gene suddenly drop. However, more interesting behavior occurs when the Hill Coefficient is 1 and below. At a Hill coefficient of 1, the equations describing the behavior of gene and mRNA repression represent standard Michaelis-Menten kinetics ([13]). Typically, a hyperbolic curve would be present to describe the generation of mRNA (and therefore protein), but because we started the model with an mRNAA concentration of 1M, there is initial oscillatory behavior. However, the oscillations quickly converge into a stable state. When the Hill coefficient drops even further to 0.1, we see a very quick emergence of a stable state at 50M for mRNA and protein. This is a predictable result: looking at the equations, with a Hill coefficient close to zero, the protein term reduces to 1, and therefore the steady state will be close to the one-half of the transcription rate.

These modulations of parameters were done with the intent to better understand the relationship between the model and the terms used. With a better understanding of the model and the oscillatory behavior of the concentrations of mRNA and protein, we hope the insight would prove useful when considering in vivo processes. Furthermore, the understanding generated here can then be used when building larger and more complex models.

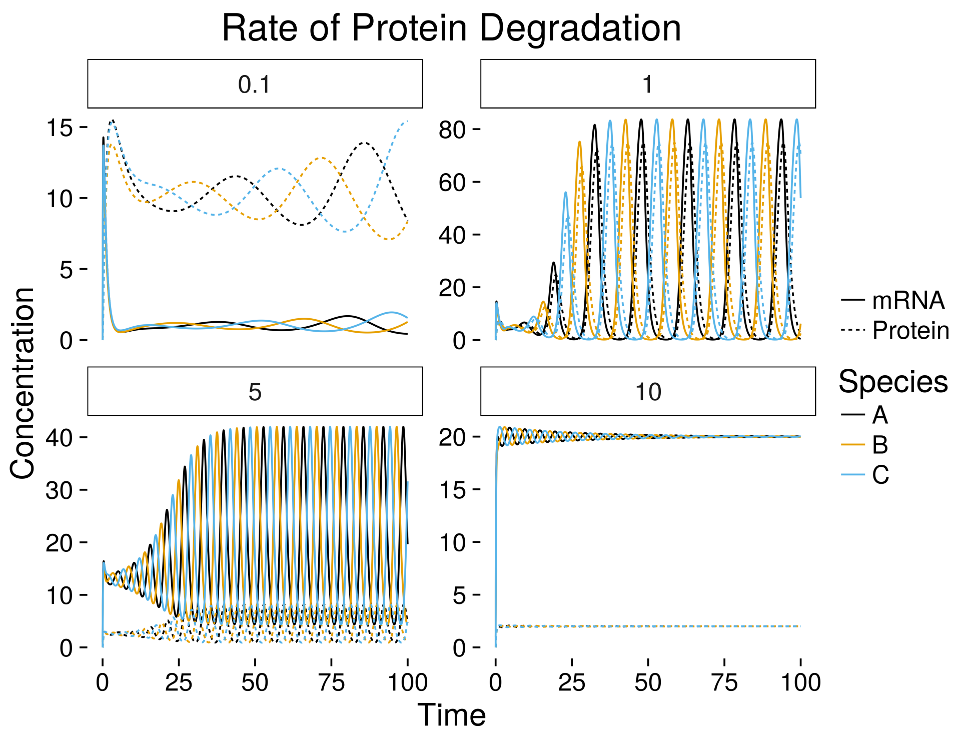

Figure 8: Steady state behavior of a single cell repressilator for varying the rates of protein degradation, k_{dp}.

However, beyond simply understanding the present model of 3 repressive genes in one cell, we wanted to look at synchronous behavior within and among cells. Primarily, we hope to understand how synchrony could be induced within cells.

To model a drug within a cell, we modified the equations to include the addition of a drug D and its rate of promotion of $mRNA_A$ (k). The rest of the equations and initial values for the model have stayed the same. Once the cell reaches steady state oscillatory behavior, we add in 100 moles of the drug. There is a jump in the concentrations of the $mRNA_A$, $mRNA_C$, $protein_A$, and $protein_C$, but $mRNA_B$ and $protein_B$ experiences repression as it is already near the minimal level of in its oscillation cycle when the drug is added (Fig. 10). Intuitively, this makes sense. The response of a single cell to an external stimulus conforms to the expectation from analyzing the circuit model: with a drug that promotes a given gene, we expect the products of that gene to experience the maximum boost and have the oscillatory pattern follow the added drug.

Knowing that a drug works to "reset" the oscillatory pattern to follow a particular species, we wanted to see if the addition of the drug would successfully place multiple cells into synchrony. We simulated 2 additional cells with the same model. However, here, we initialize one of the cells with a concentration of $mRNA_B$ at 1M and the other with a concentration of $mRNA_C$ at 1M. This will produce maximum phase lag (oscillation offset) between cells for a given gene. We again add a sizable amount of drug into the cells and the drug promotes $mRNA_A$ for each cell. If a large enough dose of drug is administered to the three cell system and its half-life is sufficiently small we

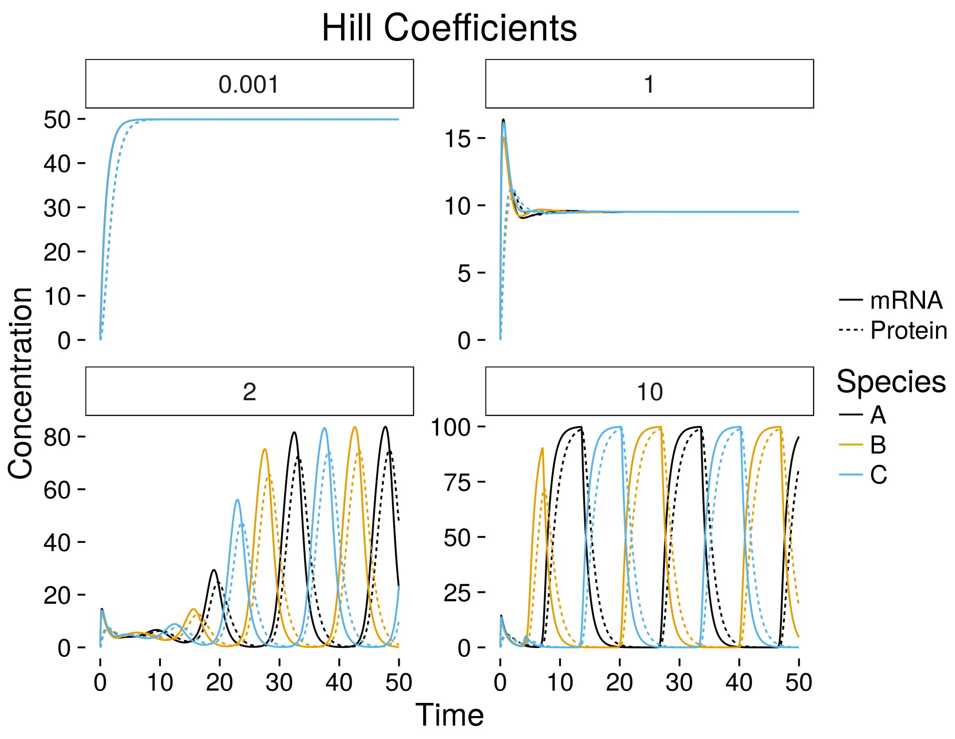

Figure 9: Steady state behavior of a single cell repressilator for varying the hill coefficients of repressor cooperativity, n.

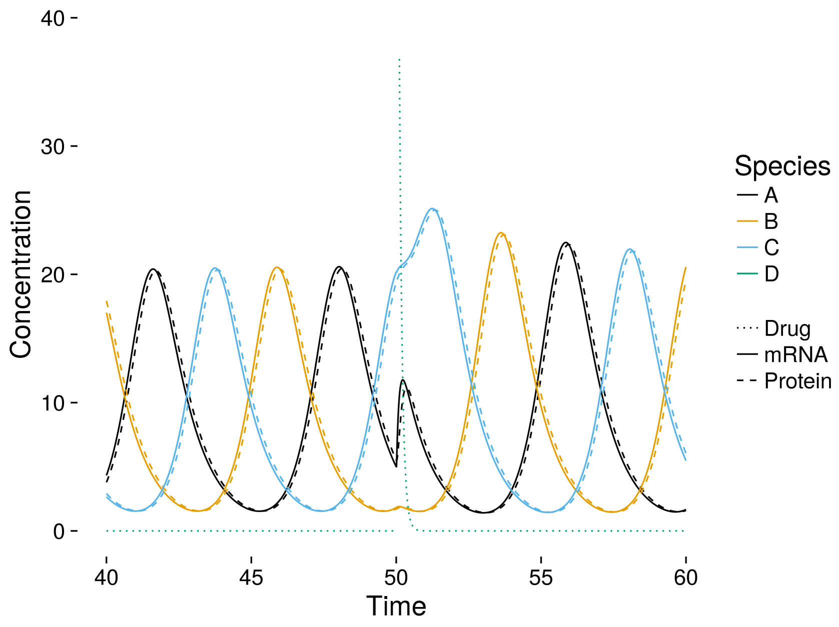

Figure 10: Transient behavior of a single cell induced with a drug at time, t=50, which promotes expression of $mRNA_A$.

can model the spike as an single pulse to each of cell. As such, we simulated adding 100 moles of the drug to each cell at the same time and observed the results.

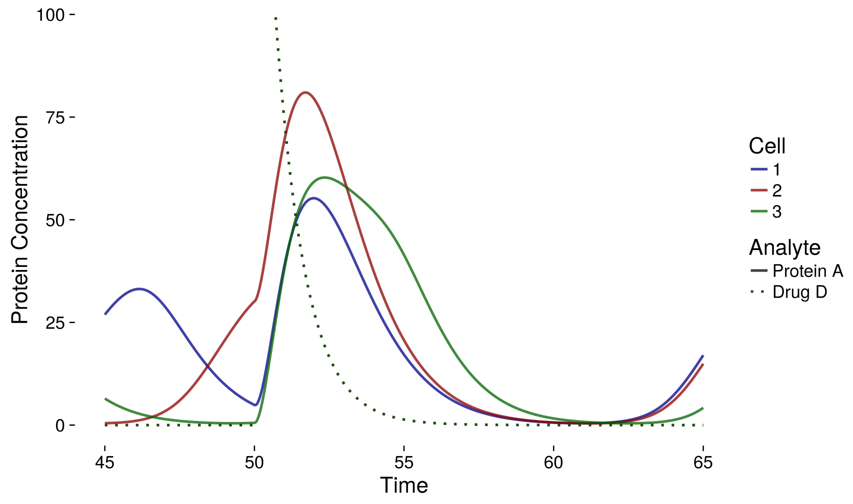

Figure 11: Initial transient behavior of three cells initially out of phase, induced with a drug which promotes expression of mRNA_A.

Each cell experiences a roughly equivalent boost in the concentration of $\mathrm{protein}_A$ (Fig. 11). Shortly after the addition of the drug, the maximum concentration of $\mathrm{protein}_A$ for each cell resembles the concentration of the species immediately after the drug addition. However, the proteins quickly return to their original oscillatory pattern, except now almost entirely in phase with each other 12 . Using this model, we were able to bring cells into synchrony, reflecting the ability of the cells to respond to drug delivery.

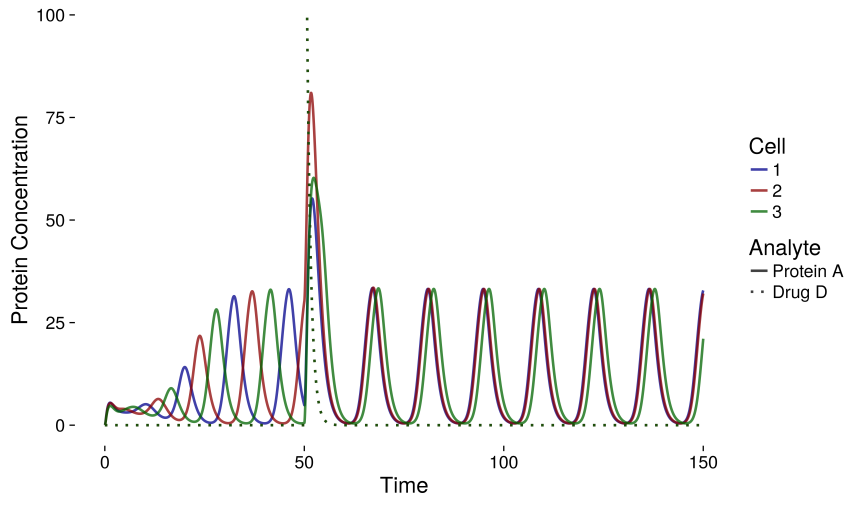

Figure 12: Steady state synchronization of three cells initially out of phase, induced with a drug which promotes expression of mRNA_A.

Knowing how cells can react in tandem and what is needed to bring them into synchrony has already been a boon for biology and medicine. Nocodazole, a cancer drug, synchronizes the cell division cycle of tumor cells, inducing particular downstream effects that cause apoptosis [14]. There are cells in the body that can synchronize themselves, as opposed to requiring a drug to do so, as in the pacemaker of the heart [15]. Chronobiology has been a highly studied field, as scientists attempt to better understand processes that exhibit oscillatory or circadian behavior, including their relationships to disease and even mental health disorders [16]. Groups of organisms as a whole are able to synchronize themselves as well; one particular example is a species of fireflies that blinks simultaneously [17]. These given cells and organisms use quorum sensing to do so; the cells must be producing their own autoinducer(s) of synchrony. Constructing these models mathematically is beyond the scope of this paper, but it is research that can most likely be done using the methods presented here. Furthermore, such networks must be expanded; genetic circuits utilize activation in addition to repression, there more genes, more cells, and cofactors, and the list of complexities continue. While we've explored a baseline model to understand repressive interactions and drug-inducible synchrony, much study remains to be done on more complex networks. We hope that as constructed models grow in size and complexity, we can begin to have a more nuanced understanding of circadian rhythms.

REFERENCES

[1] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

[2] Zachary A King, Colton J Lloyd, Adam M Feist, and Bernhard O Palsson. Next-generation genome-scale models for metabolic engineering. *Current opinion in biotechnology*, 35:23–29, 2015.

[3] Nicolas Le Novère. Quantitative and logic modelling of molecular and gene networks. *Nature Reviews Genetics*, 16(3):146–158, 2015.

[4] Javier Garcia-Bernardo and Margaret J Eppstein. Evolving modular genetic regulatory networks with a recursive, top-down approach. *Systems and Synthetic Biology*, 9(4):179–189, 2015.

[5] Nagarajan Nandagopal and Michael B Elowitz. Synthetic biology: integrated gene circuits. *science*, 333(6047):1244–1248, 2011.

[6] MAJ Roberts, RM Cranenburgh, MP Stevens, and PCF Oyston. Synthetic biology: biology by design. *Microbiology*, 159(7):1219–1220, 2013.

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

[8] Michael B Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[9] Karline Soetaert, Thomas Petzoldt, R Woodrow Setzer, et al. Solving differential equations in r: package desolve. *Journal of Statistical Software*, 33(9):1–25, 2010.

[10] N Del Buono and C Mastroserio. Explicit methods based on a class of four stage fourth order runge–kutta methods for preserving quadratic laws. *Journal of computational and applied mathematics*, 140(1):231–243, 2002.

[11] Michael R King and Nipa A Mody. Numerical and statistical methods for bioengineering. *Biomedical Instrumentation & Technology*, 2012.

[12] James N Weiss. The hill equation revisited: uses and misuses. *The FASEB Journal*, 11(11):835–841, 1997.

[13] John E Dowd and Douglas S Riggs. A comparison of estimates of michaelis-menten kinetic constants from various linear transformations. *J. biol. Chem*, 240(2):863–869, 1965.

[14] Robert J Vasquez, B Howell, AM Yvon, P Wadsworth, and L Cassimeris. Nanomolar concentrations of nocodazole alter microtubule dynamic instability in vivo and in vitro. *Molecular biology of the cell*, 8(6):973–985, 1997.

[15] Joshua I Goldhaber and John HB Bridge. Loss of intracellular and intercellular synchrony of calcium release in systolic heart failure. *Circulation: Heart Failure*, 2(3):157–159, 2009.

[16] Katharina Wulff, Silvia Gatti, Joseph G Wettstein, and Russell G Foster. Sleep and circadian rhythm disruption in psychiatric and neurodegenerative disease. *Nature Reviews Neuroscience*, 11(8):589–599, 2010.

[17] S Ripp, P Jegier, M Birmele, CM Johnson, KA Daumer, JL Garland, and GS Sayler. Linking bacteriophage infection to quorum sensing signalling and bioluminescent bioreporter monitoring for direct detection of bacterial agents. *Journal of applied microbiology*, 100(3):488–499, 2006.

4 APPENDIX

4.1 Code

4.1.1 *Single Cell Repressilator Model*

```
# 161007
# Repressilator model

library(deSolve)
library(ggplot2)
library(reshape2)

directory = "/home/fellow/Documents/Scholastic/UCSD/Fall2016/BENG221/Project"
setwd(directory)

# define a color palette
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#C

# define repressilator model
repressilator = function(times, state, parameters){
  # where a is growth rate
  # a0 = basal growth rate
  # p = protein concentration of upstream repressor
  # m = mRNA concentration

  # empty list to populate data into
  with(as.list(c(state,parameters)),{
    # calculate dervs at this time point
    dM_A =alpha_0+alpha/(1+P_C^(n))-kdm*M_A
    dP_A = kgp*M_A-kdp*P_A

    dM_B=alpha_0+alpha/(1+P_A^n)-kdm*M_B
    dP_B=kgp*M_B-kdp*P_B

    dM_C=alpha_0+alpha/(1+P_B^n)-kdm*M_C
    dP_C=kgp*M_C-kdp*P_C

    # return dervs as list
    list(c(dM_A,
           dM_B,
           dM_C,
           dP_A,
           dP_B,
           dP_C))
  })
}
```

```
# function to run the ode model
run_model = function(times, state, parameters){

  out = ode(y=state, times=times, func=repressilator,parms=parameters,method="rk4")
  out = as.data.frame(out)

  mrna = data.frame(time=out$time, A=out$M_A, B=out$M_B, C=out$M_C)
  protein = data.frame(time=out$time, A=out$P_A, B=out$P_B, C=out$P_C)

  mrna_melt = melt(mrna, id = c("time"))
  protein_melt = melt(protein, id = c("time"))

  mrna_melt$set="mRNA"
  protein_melt$set="Protein"

  recomb = rbind(mrna_melt,protein_melt)
}

###########################################
# run default repressillator model        #
###########################################

# set parameters
parameters=c(alpha_0 = 0,
             alpha=100,
             n=2,
             kgp=1,
             kdm=1,
             kdp=1)
state=c(M_A=1,
        M_B=0,
        M_C=0,
        P_A=0,
        P_B=0,
        P_C=0)
times = seq(0,10,by=0.001)

# run the model
recomb = run_model(times, state, parameters)

# plot it
p =  ggplot(recomb, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL)
ggsave("images_no_beta/161012.default_model.short_time.png",p,width=8,height=6,dpi=600)
```

```
# run the model again for longer time scale to show steady state
times = seq(0,60,by=0.01)
recomb = run_model(times, state, parameters)

# make a plot at large time scales
p = ggplot(recomb, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL)
ggsave("images_no_beta/161012.default_model.steady_state.png",p,width=8,height=6,dpi=600)


# oval plot
s2c = dcast(subset(recomb, variable=="A"), time ~ set,value.var="value")
ggplot(s2c, aes(x=mRNA,y=Protein,color=time))+
  geom_point(size=1,alpha=0.5)+
  scale_color_gradient(low="steelblue",high="red")+
  theme_bw()


##########################################
# Playing around with parameters
##########################################

###########################################
# adjust basal growth rate - alpha_0 #
###########################################

# set parameters
state=c(M_A=10,
        M_B=0,
        M_C=0,
        P_A=0,
        P_B=0,
        P_C=0)
times = seq(0,50,by=0.1)

storage = data.frame(time=numeric(),variable=character(),value=numeric(),set=character(),alpha_n

alpha_nauts = c(0.1,0.5,1,10)
for (alpha_naut in alpha_nauts){
  parameters=c( alpha_0 = alpha_naut,
                alpha=100,
                n=2,
                kgp=1,
                kdm=1,
                kdp=1)
  recomb = run_model(times, state, parameters)
  recomb$alpha_naut = alpha_naut#paste("alpha_0=",alpha_naut,sep="")
  storage = rbind(storage, recomb)
}
#storage$alpha_naut = factor(storage$alpha_naut, levels=paste("alpha_0 = ",alpha_naut,sep=""))

p = ggplot(storage, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL,title="Basal Growth Rates")+
  facet_wrap(~alpha_naut, nrow=2)#, scales="free_y")
#ggsave("images/161009.var_alpha0.x.100.10.2.png",p,width=8,height=6,dpi=600)

###########################################
# adjust promoter strength - alpha   #
###########################################

# set parameters
state=c(M_A=1,
        M_B=0,
        M_C=0,
        P_A=0,
        P_B=0,
        P_C=0)
times = seq(0,100,by=0.1)

storage = data.frame(time=numeric(),variable=character(),value=numeric(),set=character(),alpha_n

alphas = c(1,10,100,1000)
for (alpha in alphas){
  parameters=c( alpha_0 = 0,
                alpha=alpha,
                beta=10,
                n=2,
                kgp=1,
                kdm=1,
                kdp=1)
  recomb = run_model(times, state, parameters)
  recomb$alpha = alpha#paste("alpha=",alpha,sep="")
  storage = rbind(storage, recomb)
}


p = ggplot(storage, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL,title="Regulated Growth Rates")
  facet_wrap(~alpha, nrow=2, scales="free_y")
ggsave("images_no_beta/161012.alpha.png",p,width=8,height=6,dpi=600)

###########################################
# adjust decay ratio - kgp               #
###########################################

# set parameters
state=c(M_A=1,
        M_B=0,
        M_C=0,
        P_A=0,
        P_B=0,
        P_C=0)
times = seq(0,75,by=0.01)

storage = data.frame(time=numeric(),variable=character(),value=numeric(),set=character(),kgp=num

kgps = c(0.1,1,5,10)
for (kgp in kgps){
  parameters=c( alpha_0 = 0,
                alpha=100,
                beta=beta,
                n=2,
                kgp=kgp,
                kdm=1,
                kdp=1)
  recomb = run_model(times, state, parameters)
  recomb$kgp = kgp
  storage = rbind(storage, recomb)
}

p = ggplot(storage, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL,title="Rate of Protein Generati
  facet_wrap(~kgp, nrow=2, scales="free_y")
ggsave("images_no_beta/161012.kgp.png",p,width=8,height=6,dpi=600)


###########################################
# adjust decay rate - kdp                 #
###########################################

# set parameters
state=c(M_A=1,
        M_B=0,
        M_C=0,
        P_A=0,
        P_B=0,
        P_C=0)
times = seq(0,100,by=0.01)

storage = data.frame(time=numeric(),variable=character(),value=numeric(),set=character(),kdp=num

kdps = c(0.1,1,5,10)
for (kdp in kdps){
  parameters=c( alpha_0 = 0,
                alpha=100,
                beta=beta,
                n=2,
                kgp=1,
                kdm=1,
                kdp=kdp)
  recomb = run_model(times, state, parameters)
  recomb$kdp = kdp
  storage = rbind(storage, recomb)
}

p = ggplot(storage, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL,title="Rate of Protein Degradat
  facet_wrap(~kdp, nrow=2, scales="free_y")
ggsave("images_no_beta/161012.kdp.png",p,width=8,height=6,dpi=600)


###########################################
# adjust decay rate - kdm                 #
###########################################

# set parameters
state=c(M_A=1,
        M_B=0,
        M_C=0,
        P_A=0,
        P_B=0,
        P_C=0)
times = seq(0,100,by=0.01)

storage = data.frame(time=numeric(),variable=character(),value=numeric(),set=character(),kdm=num

kdms = c(0.1,1,5,10)
```

```
for (kdm in kdms){
  parameters=c( alpha_0 = 0,
                alpha=100,
                beta=beta,
                n=2,
                kgp=1,
                kdm=kdm,
                kdp=1)
  recomb = run_model(times, state, parameters)
  recomb$kdm = kdm
  storage = rbind(storage, recomb)
}

p = ggplot(storage, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL,title="Rate of mRNA Degradation
  facet_wrap(~kdm, nrow=2, scales="free_y")
ggsave("images_no_beta/161012.kdm.png",p,width=8,height=6,dpi=600)


###########################################
# adjust hill coefficient - n             #
###########################################

# set parameters
state=c(M_A=1,
        M_B=0,
        M_C=0,
        P_A=0,
        P_B=0,
        P_C=0)
times = seq(0,50,by=0.01)

storage = data.frame(time=numeric(),variable=character(),value=numeric(),set=character(),alpha_n

hills = c(0.001,1,2,10)
for (hill in hills){
  parameters=c( alpha_0 = 0,
                alpha=100,
                n=hill,
                kgp=1,
                kdm=1,
                kdp=1)
  recomb = run_model(times, state, parameters)
  recomb$hill = hill
  storage = rbind(storage, recomb)
}



p = ggplot(storage, aes(x=time,y=value,color=variable, linetype=set))+
  geom_line()+
  theme_classic(base_family="Helvetica", base_size=18)+
  scale_color_manual(values=cbbPalette)+
  labs(x="Time",y="Concentration",color="Species", linetype=NULL,title="Hill Coefficients")+
  facet_wrap(~hill, nrow=2, scales="free_y")
ggsave("images_no_beta/161012.hill.png",p,width=8,height=6,dpi=600)
```

4.1.2 *Repressilator Model with Drug Sensing*

```
# 161007
# Repressilator model with sensing feedback loop

library(deSolve)
library(ggplot2)
library(reshape2)

directory = "/home/fellow/Documents/Scholastic/UCSD/Fall2016/BENG221/Project"
setwd(directory)

# define a color palette
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#C

# define repressilator model
repressilator = function(times, state, parameters){
  # where a is growth rate
  # a0 = basal growth rate
  # p = protein concentration of upstream repressor
  # m = mRNA concentration

  # empty list to populate data into
  with(as.list(c(state,parameters)),{
    # calculate dervs at this time point

    dM_A =alpha_0+alpha/(1+P_C^(n))-kdm*M_A+ka*P_S
    dP_A = kgp*M_A-kdp*P_A

    dM_B=alpha_0+alpha/(1+P_A^n)-kdm*M_B
    dP_B=kgp*M_B-kdp*P_B

    dM_C=alpha_0+alpha/(1+P_B^n)-kdm*M_C
    dP_C=kgp*M_C-kdp*P_C

    dP_S = -kdp*(P_S)
    # return dervs as list
    list(c(dM_A,
           dM_B,
           dM_C,
           dP_A,
           dP_B,
           dP_C,
           dP_S))
  })
}


run_single_cell = function(times, state, parameters){
  out = ode(y=state, times=times, func=repressilator,parms=parameters)
  return(data.frame(out))
}


###########################################
# run default repressillator model        #
###########################################


### rerun model for each cell with final values from init run and spike of S
num_cells = 3
dt=0.01
spike_time = 60
total_time = 120
spike_level = 50

pre_times = seq(0,spike_time,by=dt)
post_times = seq(spike_time+dt,total_time, by=dt)




# set parameters
parameters=c(alpha_0 = 0,
             alpha=100,
             n=2,
             kgp=1,
             kdm=1,
             kdp=1,
             ka=1)


init_states=data.frame(M_A=c(1,0,0),
                  M_B=c(0,1,0),
                  M_C=c(0,0,1),
                  P_A=c(0,0,0),
                  P_B=c(0,0,0),
                  P_C=c(0,0,0),
                  P_S=c(0,0,0))




results = NULL

for(i in 1:num_cells){
  # grab init state for cell i
  state = setNames(as.integer(as.character(init_states[i,])), colnames(init_states))
  # run model for cell i
  out = run_single_cell(pre_times,state,parameters)
  # add cell identifier
  out$cell = i
  # store results
  if(!is.null(results)){
    results = rbind(results, out)
  } else{
    results = out
  }

  # get current state values
  cfi = out[dim(out)[1],]
  # build new state list add spike in P_S
  state = c(M_A = cfi$M_A,
            M_B = cfi$M_B,
            M_C = cfi$M_C,
            P_A = cfi$P_A,
            P_B = cfi$P_B,
            P_C = cfi$P_C,
            P_S = spike_level )

  # run model after spike
  out = run_single_cell(post_times,state,parameters)
  # add cell number to matrix
  out$cell = i
  # add it to the list of all cell state
  results = rbind(results, out)
}


# melt for plotting
rs = results[,c("time","P_A","P_S","cell")]
```

```
rsm = melt(rs, id=c("time","cell"))

# plotting
p = ggplot(rsm, aes(x=time, y=value, linetype=variable,color=as.factor(cell)))+
  geom_line()+
  labs(x="Time", y="Protein Concentration", color="Cell", linetype="Analyte")+
  scale_color_manual(values=cbbPalette)+
  theme_classic(base_family="Helvetica", base_size=18)+
  ylim(0,15)
ggsave("images/161009.3_cell_sensing.01.50.20.2.1.png",p,width=8,height=6,dpi=600)
```