Oscillations in Michaelis-Menten Systems

Hwai-Ray Tung

July 20, 2017

Abstract

Oscillations play a major role in a number of biological systems, the most notable example in biochemistry being circadian clocks. In this paper we focus on the existence of oscillations within a 2-site phosphorylation system. Previously, Wang and Sontag showed, using monotone systems theory, that the Michaelis-Menten (MM) approximation of the distributive and sequential 2-site phosphorylation system lacks oscillations. However, biological systems are generally not purely distributive; there is generally some processive behavior as well. Accordingly, this paper focuses on the MM approximation of a general sequential 2-site phosphorylation system with both processive and distributive behavior and expands on the methods of Bozeman and Morales to find conditions for the existence of oscillations. By studying the MM approximation, light may be shed on the existence of oscillations in the original system.

Introduction

Oscillations appear in a number of biological systems, including predator-prey models, the lighting up of fireflies [6], auditory hair bundles, and cytoskeletal structures. Within the context of biochemistry, much of the study of oscillations centers around genetic oscillations, which have been shown to play a role in circadian clocks, the segmentation of vertebrates, and the activity of the tumor suppressor gene p53 [4]. A simple example of genetic oscillation can be seen in Figure 1.

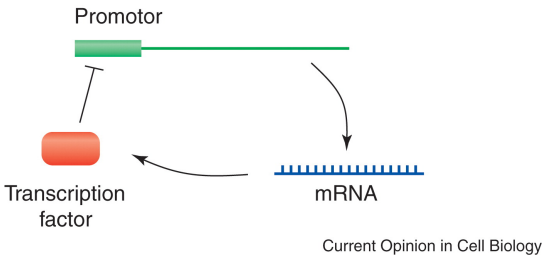

Figure 1: An example of genetic oscillation. The gene creates mRNA which creates a transcription factor. This transcription factor binds to the promoter of the gene, reducing the production of mRNA. The transcription factor is also being broken down by the cell. This leads to oscillations in the concentration of the transcription factor. Diagram taken from [4].

This paper will focus on 2-site phosphorylation systems arising from the network in (1). Phosphorylation systems can modify, activate, or deactivate proteins and have been observed in membrane receptors, protein kinases, transcription factors, cell cycle regulators, and circadian clock proteins, to name a few. The number of sites refer to the number of phosphate groups which can attach to the compound. In allowing multiple phosphate groups to attach, a compound can exhibit multiple types of behavior [5].

$$\begin{aligned} S_0 + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} S_0E \xrightarrow{k_3} S_1 + E \underset{k_5}{\overset{k_4}{\rightleftharpoons}} S_1E \xrightarrow{k_6} S_2 + E \\ S_2 + F \underset{l_2}{\overset{l_1}{\rightleftharpoons}} S_2F \xrightarrow{l_3} S_1 + F \underset{l_5}{\overset{l_4}{\rightleftharpoons}} S_1F \xrightarrow{l_6} S_0 + F \\ S_0E \xrightarrow{k_7} S_1E \\ S_2F \xrightarrow{l_7} S_1F \end{aligned} \tag{1}$$

Most of the previous work on 2-site phosphorylation systems have focused on systems with a sequential and **distributive** mechanism, displayed in (2).

$$\begin{aligned} S_0 + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} S_0E \xrightarrow{k_3} S_1 + E \underset{k_5}{\overset{k_4}{\rightleftharpoons}} S_1E \xrightarrow{k_6} S_2 + E \\ S_2 + F \underset{l_2}{\overset{l_1}{\rightleftharpoons}} S_2F \xrightarrow{l_3} S_1 + F \underset{l_5}{\overset{l_4}{\rightleftharpoons}} S_1F \xrightarrow{l_6} S_0 + F \end{aligned} \tag{2}$$

Note that (2) can be derived from (1) by setting $k_7 = l_7 = 0$. The mechanism is called sequential since the phosphate groups attach to the binding sites of the compound in a certain sequence; the phosphate group will always bind to site 1 before site 2. The mechanism is called distributive since S_1E has the choice of distributing to $S_2 + E$ or back to $S_1 + E$, and similarly for S_1F. It is unknown whether (2) exhibits oscillations, although Wang and Sontag have shown through monotone systems theory that its Michaelis-Menten (MM) approximation does not exhibit oscillations [8]. The same result has been shown by Bozeman and Morales using the more simple Dulac's criterion, which follows shortly from Green's theorem [1].

Another type of mechanism is the **processive** mechanism. A 2-site phosphorylation system with sequential and processive mechanism can be seen in (3).

$$\begin{aligned} S_0 + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} S_0E \xrightarrow{k_7} S_1E \xrightarrow{k_6} S_2 + E \\ S_2 + F \underset{l_2}{\overset{l_1}{\rightleftharpoons}} S_2F \xrightarrow{l_7} S_1F \xrightarrow{l_6} S_0 + F \end{aligned} \tag{3}$$

We note we may obtain (3) from (1) by setting rate constants $k_3 = k_4 = k_5 = l_3 = l_4 = l_5 = 0$. The mechanism is called processive since S_1E, unlike in a distributive mechanism, can only proceed to S_2+E, and similarly for S_1F. It is known that processive systems converge and do not exhibit oscillations [2][3]. As such, the MM approximation also does not yield oscillations.

In reality, 2-site phosphorylation systems are generally neither purely distributive nor purely processive. Thus, this paper will focus on the existence of oscillations in the MM approximation of (1), which is a mix of the processive and distributive systems. Section 2 details the process of obtaining the MM approximation. Section 3 gives a necessary condition for the existence of oscillations. This condition is able to again show the lack of oscillations in distributive systems as well as to prove a new result, the lack of oscillations in the mixed-mechanism network seen in (22). The code used for this project can be found in the appendices.

Michaelis-Menten System and Reduction

We would like to examine the chemical reaction network in (1). The work in this section will closely match the work of Bozeman and Morales [1]. For convenience, we will refer to the compounds S_0E, S_1E, S_2F, and S_1F as C_1, C_2, C_3, and C_4, respectively. Then, using mass action kinetics we arrive at the systems of equations

$$\frac{d[S_0]}{dt} = l_6[C_4] - k_1[S_0][E] + k_2[C_1], \tag{4a}$$

$$\frac{d[S_2]}{dt} = k_6[C_2] - l_1[S_2][F] + l_2[C_3], \tag{4b}$$

$$\frac{d[S_1]}{dt} = k_3[C_1] - k_4[S_1][E] + k_5[C_2] + l_3[C_3] + l_5[C_4] - l_4[S_1][F], \tag{4c}$$

$$\frac{d[E]}{dt} = (k_2 + k_3)[C_1] + (k_5 + k_6)[C_2] - k_1[S_0][E] - k_4[S_1][E], \tag{4d}$$

$$\frac{d[F]}{dt} = (l_2 + l_3)[C_3] + (l_5 + l_6)[C_4] - l_1[S_2][F] - l_4[S_1][F], \tag{4e}$$
$$\frac{d[S_0]}{dt} = l_6[C_4] - k_1[S_0][E] + k_2[C_1], \tag{4f}$$
$$\frac{d[C_1]}{dt} = k_1[S_0][E] - (k_2 + k_3)[C_1] - k_7[C_1], \tag{4g}$$
$$\frac{d[C_2]}{dt} = k_4[S_1][E] - (k_5 + k_6)[C_2] + k_7[C_1], \tag{4h}$$
$$\frac{d[C_3]}{dt} = l_1[S_2][F] - (l_2 + l_3)[C_3] - l_7[C_3], \tag{4i}$$
$$\frac{d[C_4]}{dt} = l_4[S_1][F] - (l_5 + l_6)[C_4] + l_7[C_3]. \tag{4j}$$

Further examining (1) gives the conservation laws

$$\begin{aligned} S_T &= [S_0] + [S_1] + [S_2] + [C_1] + [C_2] + [C_3] + [C_4], \\ E_T &= [E] + [C_1] + [C_2], \\ F_T &= [F] + [C_3] + [C_4]. \end{aligned} \tag{5}$$

We may remove S_1, E, and F through conservation equations and as such we remove (4c), (4d), and (4e) from (4). Now, we use Michaelis-Menten theory to reduce our system. We scale all variables other than S_0, S_1, S_2, S_T by ε to get

$$E_T = \varepsilon\widetilde{E_T}, \quad F_T = \varepsilon\widetilde{F_T}, \quad [C_i] = \varepsilon[\widetilde{C_i}], \quad [E] = \varepsilon[\widetilde{E}], \quad [F] = \varepsilon[\widetilde{F}], \quad \tau = \varepsilon t. \tag{6}$$

The idea behind the scaling is the assumption that the concentration of intermediates is low relative to the substrates. Substituting in the scalings from (6), we obtain

$$\begin{aligned} \frac{d[S_0]}{d\tau} &= l_6[\widetilde{C_4}] - k_1[S_0][\widetilde{E}] + k_2[\widetilde{C_1}], \\ \frac{d[S_2]}{d\tau} &= k_6[\widetilde{C_2}] - l_1[S_2][\widetilde{F}] + l_2[\widetilde{C_3}], \\ \varepsilon\frac{d[\widetilde{C_1}]}{d\tau} &= k_1[S_0][\widetilde{E}] - (k_2 + k_3)[\widetilde{C_1}] - k_7[\widetilde{C_1}], \\ \varepsilon\frac{d[\widetilde{C_2}]}{d\tau} &= k_4[S_1][\widetilde{E}] - (k_5 + k_6)[\widetilde{C_2}] + k_7[\widetilde{C_1}], \\ \varepsilon\frac{d[\widetilde{C_3}]}{d\tau} &= l_1[S_2][\widetilde{F}] - (l_2 + l_3)[\widetilde{C_3}] - l_7[\widetilde{C_3}], \\ \varepsilon\frac{d[\widetilde{C_4}]}{d\tau} &= l_4[S_1][\widetilde{F}] - (l_5 + l_6)[\widetilde{C_4}] + l_7[\widetilde{C_3}], \end{aligned} \tag{7}$$

with new conservation equations

$$\begin{aligned} S_T &= [S_0] + [S_1] + [S_2] + \varepsilon[\widetilde{C_1}] + \varepsilon[\widetilde{C_2}] + \varepsilon[\widetilde{C_3}] + \varepsilon[\widetilde{C_4}], \\ \widetilde{E_T} &= [\widetilde{E}] + [\widetilde{C_1}] + [\widetilde{C_2}], \\ \widetilde{F_T} &= [\widetilde{F}] + [\widetilde{C_3}] + [\widetilde{C_4}]. \end{aligned} \tag{8}$$

Now, we approximate the system by setting $\varepsilon = 0$ in (7) and (8). This yields the new equations for the substrates

$$\begin{aligned} \frac{d[S_0]}{d\tau} &= l_6[\widetilde{C_4}] - k_1[S_0][\widetilde{E}] + k_2[\widetilde{C_1}], \\ \frac{d[S_2]}{d\tau} &= k_6[\widetilde{C_2}] - l_1[S_2][\widetilde{F}] + l_2[\widetilde{C_3}], \end{aligned} \tag{9}$$

the new equations for the intermediates

$$\begin{aligned} 0 &= k_1[S_0][\widetilde{E}] - (k_2 + k_3)[\widetilde{C_1}] - k_7[\widetilde{C_1}], \\ 0 &= k_4[S_1][\widetilde{E}] - (k_5 + k_6)[\widetilde{C_2}] + k_7[\widetilde{C_1}], \\ 0 &= l_1[S_2][\widetilde{F}] - (l_2 + l_3)[\widetilde{C_3}] - l_7[\widetilde{C_3}], \\ 0 &= l_4[S_1][\widetilde{F}] - (l_5 + l_6)[\widetilde{C_4}] + l_7[\widetilde{C_3}], \end{aligned} \tag{10}$$

and the new conservation equations

$$\begin{aligned} S_T &= [S_0] + [S_1] + [S_2], \\ \widetilde{E_T} &= [\widetilde{E}] + [\widetilde{C_1}] + [\widetilde{C_2}], \\ \widetilde{F_T} &= [\widetilde{F}] + [\widetilde{C_3}] + [\widetilde{C_4}]. \end{aligned} \tag{11}$$

From (10) we obtain the equations

$$\begin{aligned} [\widetilde{C_1}] &= \frac{k_1}{k_2+k_3+k_7}[S_0][\widetilde{E}], & [\widetilde{C_3}] &= \frac{l_1}{l_2+l_3+l_7}[S_2][\widetilde{F}], \\ [\widetilde{C_2}] &= \frac{k_4[S_1][\widetilde{E}] + k_7[\widetilde{C_1}]}{k_5+k_6} & [\widetilde{C_4}] &= \frac{l_4[S_1][\widetilde{F}] + l_7[\widetilde{C_3}]}{l_5+l_6} \\ &= \frac{k_4[S_1][\widetilde{E}]}{k_5+k_6} + \frac{k_1k_7[S_0][\widetilde{E}]}{(k_5+k_6)(k_2+k_3+k_7)}, & &= \frac{l_4[S_1][\widetilde{F}]}{l_5+l_6} + \frac{l_1l_7[S_2][\widetilde{F}]}{(l_5+l_6)(l_2+l_3+l_7)}. \end{aligned} \tag{12}$$

Substituting the expressions for $[\widetilde{C_i}]$ in 12 into (11) gives

$$\begin{aligned} \widetilde{E_T} &= [\widetilde{E}]\left[1 + \frac{k_1}{k_2+k_3+k_7}[S_0] + \frac{k_4[S_1]}{k_5+k_6} + \frac{k_1k_7[S_0]}{(k_5+k_6)(k_2+k_3+k_7)}\right], \\ \widetilde{F_T} &= [\widetilde{F}]\left[1 + \frac{l_1}{l_2+l_3+l_7}[S_2] + \frac{l_4[S_1]}{l_5+l_6} + \frac{l_1l_7[S_2]}{(l_5+l_6)(l_2+l_3+l_7)}\right]. \end{aligned} \tag{13}$$

Substituting (12) and (13) into (9) and assigning some new variables in (15) yields the system

$$\begin{aligned} \frac{d[S_0]}{d\tau} &= \frac{a_1[S_1][\widetilde{F_T}]}{1+c_2[S_1]+d_1[S_2]} + \frac{a_2[S_2][\widetilde{F_T}]}{1+c_2[S_1]+d_1[S_2]} - \frac{a_3[S_0][\widetilde{E_T}]}{1+b_1[S_0]+c_1[S_1]}, \\ \frac{d[S_2]}{d\tau} &= \frac{a_4[S_1][\widetilde{E_T}]}{1+b_1[S_0]+c_1[S_1]} + \frac{a_5[S_0][\widetilde{E_T}]}{1+b_1[S_0]+c_1[S_1]} - \frac{a_6[S_2][\widetilde{F_T}]}{1+c_2[S_1]+d_1[S_2]}, \end{aligned} \tag{14}$$

where

$$\begin{aligned} b_1 &= \frac{k_1(k_5+k_6+k_7)}{(k_2+k_3+k_7)(k_5+k_6)}, & d_1 &= \frac{l_1(l_5+l_6+l_7)}{(l_2+l_3+l_7)(l_5+l_6)}, \\ c_1 &= \frac{k_4}{k_5+k_6}, & c_2 &= \frac{l_4}{l_5+l_6}, \\ a_1 &= \frac{l_4l_6}{l_5+l_6}, & a_4 &= \frac{k_4k_6}{k_5+k_6}, \\ a_2 &= \frac{l_1l_6l_7}{(l_2+l_3+l_7)(l_5+l_6)}, & a_5 &= \frac{k_1k_6k_7}{(k_2+k_3+k_7)(k_5+k_6)}, \\ a_3 &= \frac{k_1(k_3+k_7)}{(k_2+k_3+k_7)}, & a_6 &= \frac{l_1(l_3+l_7)}{(l_2+l_3+l_7)}, \end{aligned} \tag{15}$$

with conservation equation

$$S_T = [S_0] + [S_1] + [S_2]. \tag{16}$$

From (16), we obtain $[S_1] = S_T - [S_0] - [S_2]$. Substituting this into (14) yields a system of only two variables, as seen below. This system is our **MM approximation**.

$$\begin{aligned} \frac{d[S_0]}{d\tau} &= \frac{(a_1(S_T-[S_0]-[S_2])+a_2[S_2])[\widetilde{F_T}]}{1+c_2(S_T-[S_0]-[S_2])+d_1[S_2]} - \frac{a_3[S_0][\widetilde{E_T}]}{1+b_1[S_0]+c_1(S_T-[S_0]-[S_2])}, \\ \frac{d[S_2]}{d\tau} &= \frac{(a_4(S_T-[S_0]-[S_2])+a_5[S_0])[\widetilde{E_T}]}{1+b_1[S_0]+c_1(S_T-[S_0]-[S_2])} - \frac{a_6[S_2][\widetilde{F_T}]}{1+c_2(S_T-[S_0]-[S_2])+d_1[S_2]}. \end{aligned} \tag{17}$$

Necessary Condition for Oscillations in the MM System

To analyze (17), we use Dulac's Criterion, which is also called the Bendixson-Dulac theorem and Bendixson's criterion.

Theorem 1. *(Dulac's Criterion) Let $\frac{dx}{dt} = f(x,y)$ and $\frac{dy}{dt} = g(x,y)$ be a system of ODEs defined on a simply connected region D of $\mathbb{R}^2$. If $\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y}$ is always positive or always negative on the region D, the system does not exhibit oscillations contained in region D.*

We are now ready to prove the main result of this paper, Theorem 2.

Theorem 2. *Assume $a_3 + a_4 > 0$ and $a_1 + a_6 > 0$. In order for the MM approximation (17) to exhibit oscillations, one of the following must be true:*

$$(a_5c_1 - a_4b_1 - a_3c_1)S_T \geq a_3 + a_4$$

or

$$(a_2c_2 - a_1d_1 - a_6c_2)S_T \geq a_1 + a_6.$$

Proof. We now apply Dulac's Criterion to (17) with $f([S_0],[S_2]) \coloneqq \frac{d[S_0]}{d\tau}$, $g([S_0],[S_2]) \coloneqq \frac{d[S_2]}{d\tau}$, and $H \coloneqq \frac{\partial f}{\partial [S_0]} + \frac{\partial g}{\partial [S_2]}$ on the region $D \coloneqq \{([S_0],[S_2]) \in \mathbb{R}^2 \quad | \quad [S_0] \geq 0, [S_2] \geq 0, [S_0] + [S_2] \leq S_T\}$. It is straightforward to check that

$$H(0,0) = -\widetilde{E_T}\frac{a_3c_1S_T + a_3 + a_4}{(c_1S_T+1)^2} - \widetilde{F_T}\frac{a_6c_2S_T + a_1 + a_6}{(c_2S_T+1)^2} < 0. \tag{18}$$

Then by Theorem 1, for our reduced system to have oscillations, there must be a point $([S_0],[S_2])$ in our domain such that $H([S_0],[S_2]) \geq 0$. Simplifying, we find that $H([S_0],[S_2]) \geq 0$ is equivalent to the following inequality:

$$\begin{aligned}&\widetilde{E_T}(c_2[S_1] + d_1[S_2] + 1)^2((a_5c_1 - a_4b_1)[S_0] + a_3c_1[S_2] - a_3c_1S_T - a_3 - a_4)\\ &+\widetilde{F_T}(c_1[S_1] + b_1[S_0] + 1)^2(a_6c_2[S_0] + (a_2c_2 - a_1d_1)[S_2] - a_6c_2S_T - a_1 - a_6) \geq 0.\end{aligned} \tag{19}$$

In order for (19) to hold it is necessary for one of the two following expressions to be positive.

$$(a_5c_1 - a_4b_1)[S_0] + a_3c_1[S_2] - a_3c_1S_T - a_3 - a_4, \tag{20a}$$
$$a_6c_2[S_0] + (a_2c_2 - a_1d_1)[S_2] - a_6c_2S_T - a_1 - a_6. \tag{20b}$$

For our domain D, we note that the maxima of (20a) and (20b), respectively, are

$$(\max(a_5c_1 - a_4b_1, a_3c_1) - a_3c_1)S_T - a_3 - a_4, \tag{21a}$$
$$(\max(a_2c_2 - a_1d_1, a_6c_2) - a_6c_2)S_T - a_1 - a_6. \tag{21b}$$

Then, in order for (20a) or (20b) to be positive it is necessary that $\max(a_5c_1 - a_4b_1, a_3c_1) = a_5c_1 - a_4b_1$ or $\max(a_2c_2 - a_1d_1, a_6c_2) = a_2c_2 - a_1d_1$. Thus, it is necessary that

$$(a_5c_1 - a_4b_1 - a_3c_1)S_T - a_3 - a_4 \geq 0$$

or

$$(a_2c_2 - a_1d_1 - a_6c_2)S_T - a_1 - a_6 \geq 0,$$

showing our desired result. The code assisting with the algebra of this proof can be found in Appendix A. □

The assumptions stated in Theorem 2 are biologically reasonable; if $a_3 + a_4 = 0$, then $k_1 = 0$ or $k_3 = k_7 = 0$, which would result in S_0 not being able to become S_2. Analysis on the condition $a_1 + a_6 > 0$ follows similarly.

With Theorem 2 we obtain the following corollaries on a purely distributive system, a purely processive system, and another subsystem known as the **mixed-mechanism network**, seen in (22). First, we recover the earlier result that the distributive network does not exhibit oscillations [1][8].

Corollary 3. *The MM approximation of the distributive model (2) does not admit oscillations.*

Proof. In the distributive model (2), the mass action constants k_7, l_7 are set to 0 while the others are positive, which implies that $a_2 = a_5 = 0$. Substituting these values into the inequalities of Theorem 2 gives $-(a_4b_1 + a_3c_1)S_T \geq a_3 + a_4$ and $-(a_1d_1 + a_6c_2)S_T \geq a_1 + a_6$. Note that the left hand sides $-(a_4b_1 + a_3c_1)S_T$ and $-(a_1d_1 + a_6c_2)S_T$ are negative while the right hand sides $a_3 + a_4$ and $a_1 + a_6$ are positive. Thus, neither inequality holds and the system has no oscillations. □

Next, we show that the processive model does not exhibit oscillations. This is to be expected from work by Conradi & Shiu and Eithun & Shiu, who show that processive multisite phosphorylation networks do not exhibit oscillations [2][3]. As the original does not exhibit oscillations, it is reasonable to expect the MM approximation to nto exhibit oscillations as well.

Corollary 4. *The MM approximation of the processive model (3) does not admit oscillations.*

Proof. In the processive model (3), the mass action constants $k_3, k_4, k_5, l_3, l_4, l_5$ are set to 0 while the others are positive, which implies that $c_1 = c_2 = a_1 = a_4 = 0$. Substituting these values into the inequalities of Theorem 2 gives $0 \geq a_3$ and $0 \geq a_6$. Neither inequality holds as by assumption a_3 and a_6 are positive. □

The next corollary gives a new result for the MM approximation of the mixed-mechanism network, seen in (22).

$$S_0 + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} S_0E \xrightarrow{k_7} S_1E \xrightarrow{k_6} S_2 + E \tag{22a}$$

$$S_2 + F \underset{l_2}{\overset{l_1}{\rightleftharpoons}} S_2F \xrightarrow{l_3} S_1 + F \underset{l_5}{\overset{l_4}{\rightleftharpoons}} S_1F \xrightarrow{l_6} S_0 + F \tag{22b}$$

The mixed-mechanism network can be obtained from our original system (1) by setting $k_3 = k_4 = k_5 = l_7 = 0$. It is called the mixed-mechanism network since (22a) is processive while (22b) is distributive. Corollary 5 is of particular interest because Suwanmajo and Krishnan showed that the original mixed-mechanism system does exhibit oscillations [7], even though corollary 5 proves that the MM approximation of the mixed-mechanism does not admit oscillations. This serves as a good reminder that the MM approximation is an approximation, and information is lost in exchange for ease of algebra.

Corollary 5. *The MM approximation of the mixed-mechanism network does not admit oscillations.*

Proof. In (22), the mass action constants k_3, k_4, k_5, l_7 are set to 0 while the others are positive, which implies that $c_1 = a_2 = a_4 = 0$. Substituting these values into the inequalities of Theorem 2 gives $0 \geq a_3$ and $-(a_1d_1 + a_6c_2)S_T \geq a_1 + a_6$. Note that the left hand sides 0 and $-(a_1d_1 + a_6c_2)S_T$ are not positive while the right hand sides a_3 and $a_1 + a_6$ are positive. Thus, neither inequality holds and the system has no oscillations. □

Discussion

The goal of this paper was to investigate oscillations in the general sequential 2-site phosphorylation network with both processive and distributive elements. We began by using mass action kinetics to write our network as a system of ODEs. We then made assumptions on the concentration of intermediates to obtain the MM approximation. Using Theorem 1 (Dulac's criterion), we obtained a necessary condition for oscillations in the MM approximation, Theorem 2. Using Theorem 2, we were able to confirm the lack of oscillations in the distributive network and processive network as well as show for the first time that the MM approximation of the mixed-mechanism network exhibits no oscillations.

The natural next step is to ask whether there ever exists oscillations in the MM approximation of the general case. Numerical experimentation suggests not. In my code in Appendix B, I randomly substituted the integers 1 through 10 into the parameters in (15) and the conservation law constants in (5). Upon substitution, my code calculates the sign of the minimum and maximum values the sum in Dulac's criterion can take, thus determining whether Dulac's criterion precludes oscillations for a parameter set. This left roughly 5% of parameter sets that potentially exhibit oscillations.

I then generated 300 parameter sets that potentially exhibited oscillations and performed a vector plot. A point was colored blue, yellow, green, or red when the direction vector was pointing in quadrant I, II, III, or IV, respectively. One example can be seen in Figure 2. Only 3 of the 300 figures contained all 4 colors, which is necessary for oscillations. As such, it would seem at the least rare for oscillations to occur in the MM approximation.

As no confirmed instances of oscillations have been found, it may be possible that oscillations are not obtainable. One potential route to a proof of no oscillations in the general case would be to generalize the work of Wang and Sontag [8]. The main theorem of their paper, which was used to show the lack of oscillations in the distributive MM approximation, relies on 7 conditions holding. The conditions and theorem can be found in section 2 of their paper. The first through fifth have been shown to hold for the general case and the seventh is believed to hold; see Appendix D for code on the first through fifth. The

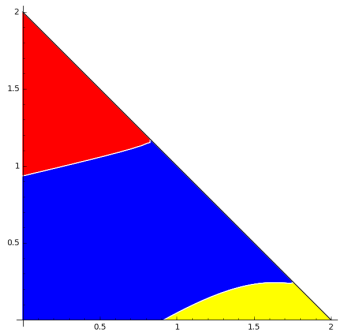

Figure 2: The colored vector plot with parameters $a_1 = 2, a_2 = 9, a_3 = 3, a_4 = 6, a_5 = 9, a_6 = 8, b_1 = 3, c_1 = 9, c_2 = 4, d_1 = 1, \widetilde{E_T} = 6, \widetilde{F_T} = 3, S_T = 2$. The x and y axes are $[S_0]$ and $[S_2]$ respectively. The points of a vector plot are colored blue, yellow, green, or red when the direction vector is pointing in quadrant I, II, III, or IV, respectively. As only 3 of the colors are exhibited, this parameter set does not admit oscillations. This plot was generated through Sage and can be found in Appendix C.

sixth condition $A6$ is where the majority of the difficulty is expected to lie. As seen in Figure 3, finding a cone to work over that results in a proof of $A6$ may prove challenging, or even impossible.

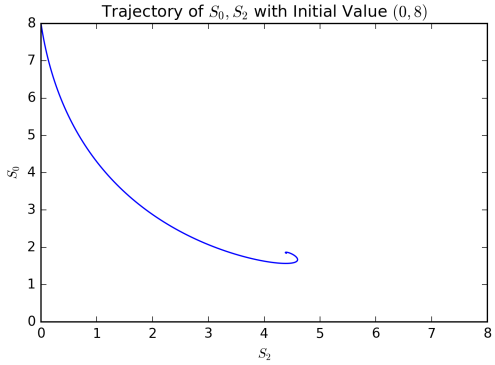

Figure 3: The plot of a solution with initial value $(0, 8)$ and parameters $a_1 = 5, a_2 = 1, a_3 = 6, a_4 = 6, a_5 = 8, a_6 = 10, b_1 = 1, c_1 = 10, c_2 = 3, d_1 = 10, \widetilde{E_T} = 3, \widetilde{F_T} = 8, S_T = 8$. The x and y axes are $[S_0]$ and $[S_2]$ respectively. The trajectory leaves the initial value and heads for the fixed point near $(4.5, 2)$. The way the trajectory overshoots the fixed point, then doubles back may make it difficult to apply monotone systems theory. This plot was generated through Python and can be found in Appendix E.

Another question arising from this project is the legitimacy of using the MM approximation. As we saw with the mixed-mechanism network, lack of oscillations in the MM approximation does not imply lack of oscillations in the original network. A more obvious way in which the MM approximation differs from the original is in the apparent violation of conservation laws. As seen in Figure 4, despite the conservation law $[S_0] + [S_1] + [S_2] = S_T = 2$ which implies $[S_0] + [S_2] \leq S_T = 2$, the vector plot indicates many solutions which exit the domain. In other words, behavior near the $[S_1] = 0$ diagonal in Figure 4 is not biologically realistic. Such violations are not present in the original system. As such, further research into what information is preserved when performing MM approximation may prove interesting.

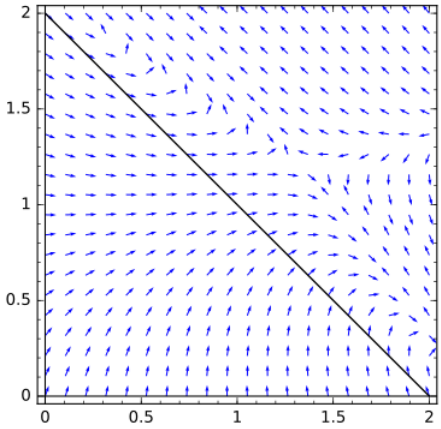

Figure 4: The vector plot with parameters $a_1 = 2, a_2 = 9, a_3 = 3, a_4 = 6, a_5 = 9, a_6 = 8, b_1 = 3, c_1 = 9, c_2 = 4, d_1 = 1, \widetilde{E_T} = 6, \widetilde{F_T} = 3, S_T = 2$. The x and y axes are $[S_0]$ and $[S_2]$ respectively. We can see solutions crossing the boundary $S_0 + S_2 = S_T = 2$. This plot was generated through Sage and can be found in Appendix C.

Acknowledgements

This project would not have been possible without the mentorship of Dr. Anne Shiu and the support of Jonathan Tyler. This research was conducted with funding from the NSF (DMS-1460766) at the REU program of Texas A&M University, Summer 2017.

References

[1] Luna Bozeman and Adriana Morales. "No oscillations in the Michaelis-Menten approximation of the dual futile cycle under a sequential and distributive mechanism". In: *SIAM Undergraduate Research Online* 10 (2017), pp. 21–28.

[2] Carsten Conradi and Anne Shiu. "A Global Convergence Result for Processive Multisite Phosphorylation Systems". In: *Bulletin of Mathematical Biology* 77.1 (Jan. 2015), pp. 126–155. ISSN: 1522-9602. DOI: 10.1007/s11538-014-0054-4. URL: http://dx.doi.org/10.1007/s11538-014-0054-4.

[3] Mitchell Eithun and Anne Shiu. "An all-encompassing global convergence result for processive multisite phosphorylation systems". In: *Mathematical Biosciences* 291 (2017), pp. 1–9. ISSN: 0025-5564. DOI: http://dx.doi.org/10.1016/j.mbs.2017.05.006. URL: http://www.sciencedirect.com/science/article/pii/S0025556417300160.

[4] Karsten Kruse and Frank Jülicher. "Oscillations in cell biology". In: *Current Opinion in Cell Biology* 17.1 (2005), pp. 20–26. ISSN: 0955-0674. DOI: http://dx.doi.org/10.1016/j.ceb.2004.12.007. URL: http://www.sciencedirect.com/science/article/pii/S0955067404001759.

[5] C. Salazar and T. Höfer. "Multisite protein phosphorylation - from molecular mechanisms to kinetic models." In: *FEBS Journal* 276.12 (2009), pp. 3177-3198.

[6] S.H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Studies in Nonlinearity. Avalon Publishing, 2014. ISBN: 9780813349107. URL: https://books.google.com/books?id=aMrSDQAAQBAJ.

[7] Thapanar Suwanmajo and J. Krishnan. "Mixed mechanisms of multi-site phosphorylation". In: *Journal of The Royal Society Interface* 12.107 (2015). ISSN: 1742-5689. DOI: 10.1098/rsif.2014.1405. eprint: http://rsif.royalsocietypublishing.org/content/12/107/20141405.full.pdf. URL: http://rsif.royalsocietypublishing.org/content/12/107/20141405.

[8] Liming Wang and Eduardo D. Sontag. "Singularly Perturbed Monotone Systems and an Application to Double Phosphorylation Cycles". In: *Journal of Nonlinear Science* 18.5 (Oct. 2008), pp. 527–550. ISSN: 1432-1467. DOI: 10.1007/s00332-008-9021-2. URL: http://dx.doi.org/10.1007/s00332-008-9021-2.

Appendix A

This appendix contains the Python code used to work through the algebra of Theorem 2.

```
import sympy as sp
import datetime

from IPython.display import display

print "START", datetime.datetime.now()

a1, a2, a3, a4, a5, a6, b1, c1, c2, d1, S0, S2, ET, FT, ST, l0, l2, y = sp.
    var('a1 a2 a3 a4 a5 a6 b1 c1 c2 d1 S0 S2 ET FT ST l0 l2 y')
S1 = ST - S0 - S2

#our MM approx equations
dS0dt = a1*S1*FT/(1+c2*S1+d1*S2) + a2*S2*FT/(1+c2*S1+d1*S2) - a3*S0*ET/(1+
    b1*S0+c1*S1)
dS2dt = a4*S1*ET/(1+b1*S0+c1*S1) + a5*S0*ET/(1+b1*S0+c1*S1) - a6*S2*FT/(1+
    c2*S1+d1*S2)

print"_____"

#our sum of interest in Dulac
s = (sp.diff(dS2dt, S2) + sp.diff(dS0dt, S0))

#For origin case
sOrigin = s.subs({S0: 0, S2:0})
sOriginET = sOrigin.coeff(ET, 1)
sOriginFT = sOrigin.coeff(FT, 1)
sOriginET = sp.together(sOriginET)
sOriginFT = sp.together(sOriginFT)
print "Numerator of Coefficient of ET for origin"
display(sp.expand(sp.numer(sOriginET)))
print"_____"
print "Numerator of Coefficient of FT for origin"
display(sp.expand(sp.numer(sOriginFT)))
print"_____"
#Note s = ET*sOriginET + FT*sOriginFT

#For General Case
s = sp.together(s)
sNum, sDen = sp.fraction(s)
sNumET = sNum.coeff(ET, 1)
sNumFT = sNum.coeff(FT, 1)
print"_____"
print "Numerator of Coefficient of ET in general case"
display(sNumET.factor())
print"_____"
print "Numerator of Coefficient of FT in general case"
display(sNumFT.factor())
print"_____"
#Note s = (ET*sNumET + FT*sNumFT)/sDen

print "END", datetime.datetime.now()

#
    ================================================================================

# #For Showing Distributive
```

```
# s = (sp.diff(dS2dt, S2) + sp.diff(dS0dt, S0))
# s = s.subs({a2:0, a5:0})
# sET = s.coeff(ET, 1)
# sFT = s.coeff(FT, 1)
# sET = sp.together(sET)
# sFT = sp.together(sFT)
# display(sp.expand(sp.numer(sET)))
# display(sp.expand(sp.numer(sFT)))
#
    ================================================================================


#
    ================================================================================

## For showing Processive
# s = (sp.diff(dS2dt, S2) + sp.diff(dS0dt, S0))
# s = s.subs({c1:0, c2:0, a1:0, a4:0})
# sET = s.coeff(ET, 1)
# sFT = s.coeff(FT, 1)
# sET = sp.together(sET)
# sFT = sp.together(sFT)
# display(sET)
# display(sFT)
#
    ================================================================================
```

Appendix B

This appendix contains the Python code used to generate 300 parameter sets which possibly exhibit oscillations under Theorem 1 (Dulac's Criterion).

```
import sympy as sp
import csv
import random
import datetime

#Code generate 300 examples that can possibly oscillate from Dulac
#General idea is generate random parameters, then find min and max
#    of sum in Dulac and if the signs of min and max are different

#check if sign changes over the domain
def sameSign(expression, ST, S0, S2):
    boundaryBounds = getMaxBoundary(expression, ST, S0, S2)
    interiorBounds = getMaxInterior(expression, ST, S0, S2)
    return min(boundaryBounds[0], interiorBounds[0]) == max(boundaryBounds
        [1], interiorBounds[1])

#check the signs on the boundaries of the domain
def getMaxBoundary(expression, ST, S0, S2):
    case1 = getMaxBoundaryCase(expression, 0, ST, S0, 0, S2)
    case2 = getMaxBoundaryCase(expression, 0, ST, S2, 0, S0)
    case3 = getMaxBoundaryCase(expression, 0, ST, S2, ST - S0, S0)
    return [min(case1[0], case2[0], case3[0]), max(case1[1], case2[1],
        case3[1])]

#check the signs on a boundary
def getMaxBoundaryCase(expression, bound1, bound2, varToSubOut, replaceWith
    , otherVar):
    case = expression.subs({varToSubOut : replaceWith})
    diff = sp.diff(case, otherVar)
    sols = sp.solve(diff, otherVar)
    b1, b2 = sp.sign(case.subs({otherVar : bound1})), sp.sign(case.subs({
        otherVar : bound2}))
    maxCase = max(b1, b2)
    minCase = min(b1, b2)
    for sol in sols:
        if sol.is_real and sol > bound1 and sol < bound2:
            maxCase = max(maxCase, sp.sign(case.subs({otherVar : sol})))
            minCase = min(minCase, sp.sign(case.subs({otherVar : sol})))
    return [minCase, maxCase]

#check for sign changes in interior of boundary
def getMaxInterior(expression, ST, S0, S2):
    dS0 = sp.numer(sp.together(sp.diff(expression, S0)))
    dS2 = sp.numer(sp.together(sp.diff(expression, S2)))
    sols = getSolsWithResultant(dS0, dS2, ST)
    maxCase = sp.sign(expression.subs({S0 : 0, S2 : 0}))
    minCase = maxCase
    for sol in sols:
        if sol[0].is_real and sol[1].is_real and sol[0] + sol[1] < ST and
            sol[0] > 0 and sol[1] > 0:
            maxCase = max(maxCase, sp.sign(expression.subs({S0 : sol[0], S2
                : sol[1]})))
            minCase = min(minCase, sp.sign(expression.subs({S0 : sol[0], S2
                : sol[1]})))
    return [minCase, maxCase]
```

```
#use resultants to get solutions to exp1, exp2 that are in domain
def getSolsWithResultant(exp1, exp2, ST1):
    sols = []
    resS0 = sp.resultant(exp1, exp2, S0)
    resS2 = sp.resultant(exp1, exp2, S2)
    S2sols = sp.solve(resS0, S2)
    S0sols = sp.solve(resS2, S0)
    for x in S0sols:
        for y in S2sols:
            if x>0 and y>0 and x+y <ST1 and exp1.subs({S0 : x, S2 : y}) ==
                0 and exp2.subs({S0 : x, S2 : y}) == 0:
                sols.append([x, y])
    return sols

print "START", datetime.datetime.now()

a1, a2, a3, a4, a5, a6, b1, c1, c2, d1, S0, S2, ET, FT, ST = sp.symbols('a1
    a2 a3 a4 a5 a6 b1 c1 c2 d1 S0 S2 ET FT ST', real = True)
S1 = ST - S0 - S2

#the MM approx equations
dS0dt = a1*S1*FT/(1+c2*S1+d1*S2) + a2*S2*FT/(1+c2*S1+d1*S2) - a3*S0*ET/(1+
    b1*S0+c1*S1)
dS2dt = a4*S1*ET/(1+b1*S0+c1*S1) + a5*S0*ET/(1+b1*S0+c1*S1) - a6*S2*FT/(1+
    c2*S1+d1*S2)

dfdS0 = sp.diff(dS0dt, S0)
fgdS2 = sp.diff(dS2dt, S2)

#the sum of interest for Dulac
s = dfdS0 + fgdS2

keys = [a1, a2, a3, a4, a5, a6, b1, c1, c2, d1, ET, FT, ST]

counterexamples = []

while len(counterexamples)<300:
    #generate random parameters and plug into equation s
    params = [random.randint(1, 10) for j in range(len(keys))]
    paramDict = dict(zip(keys, params))
    example = s.subs(paramDict)

    try:
        if not sameSign(example, paramDict[ST], S0, S2):
            print "Couterexample at", params
            counterexamples.append(params)
            if len(counterexamples)%10 == 0:
                print "Currently at", len(counterexamples)
    #every once in a while an equation can't be solved, making a type error
    except TypeError:
        pass

with open('counters.csv', 'wb') as f:
    writer = csv.writer(f)
    writer.writerows(counterexamples)

print "End at", datetime.datetime.now()
```

Appendix C

This appendix contains the Sage code used to produce Figures 2 and 4.

```
import datetime
import sympy as sp

print "START", datetime.datetime.now()

a1, a2, a3, a4, a5, a6, b1, c1, c2, d1, S0, S2, ET, FT, ST, l0, l2, y = var
    ('a1 a2 a3 a4 a5 a6 b1 c1 c2 d1 S0 S2 ET FT ST l0 l2 y')
S1 = ST - S0 - S2

dS0dt = a1*S1*FT/(1+c2*S1+d1*S2) + a2*S2*FT/(1+c2*S1+d1*S2) - a3*S0*ET/(1+
    b1*S0+c1*S1)
dS2dt = a4*S1*ET/(1+b1*S0+c1*S1) + a5*S0*ET/(1+b1*S0+c1*S1) - a6*S2*FT/(1+
    c2*S1+d1*S2)

s = dS0dt + dS2dt

#Assign values to parameters and plug in
keys = [a1, a2, a3, a4, a5, a6, b1, c1, c2, d1, ET, FT, ST]
params = [2, 9, 3, 6, 9, 8, 3, 9, 4, 1, 6, 3, 2]
paramDict = dict(zip(keys, params))
dS0 = dS0dt.substitute(paramDict)
dS2 = dS2dt.substitute(paramDict)

startingx = 0
startingy = 0

print "Arrow Direction of vector plot. Triangle is domain"

p=region_plot([dS2> 0, dS0>0, S0+S2<=paramDict[ST]],(S0,startingx,paramDict
    [ST]),(S2,startingy,paramDict[ST]), incol = 'blue', axes_labels=['$S_0$
    Concentration','$S_2$ Concentration'], title = 'Colored Vector Plot of $
    (S_0, S_2)$')
p+=region_plot([dS2< 0, dS0>0, S0+S2<=paramDict[ST]],(S0,startingx,
    paramDict[ST]),(S2,startingy,paramDict[ST]), incol = 'red')
p+=region_plot([dS2> 0, dS0<0, S0+S2<=paramDict[ST]],(S0,startingx,
    paramDict[ST]),(S2,startingy,paramDict[ST]), incol = 'yellow')
p+=region_plot([dS2< 0, dS0<0, S0+S2<=paramDict[ST]],(S0,startingx,
    paramDict[ST]),(S2,startingy,paramDict[ST]), incol = 'green')
p+=region_plot([dS2== 0, dS0==0, S0+S2<=paramDict[ST]],(S0,startingx,
    paramDict[ST]),(S2,startingy,paramDict[ST]), incol = 'gold')
p+=plot(paramDict[ST]-S0, (S0, startingx, paramDict[ST]), color = 'black')
p.show(axes = "True")
print "Blue   is  NE (Quadrant I)"
print "Red    is  SE (Quadrant IV)"
print "Yellow is  NW (Quadrant II)"
print "Green  is  SW (Quadrant III)"

print"-----------------------------------------"

print "Region where Dulac sum is positive is green. Negative or 0 is white"
p3 = region_plot(diff(dS0, S0) + diff(dS2, S2) > 0, (S0,0,paramDict[ST]),(
    S2,0,paramDict[ST]), incol = 'green')
p3 +=plot(paramDict[ST]-S0, (S0, startingx, paramDict[ST]), color = 'black'
    )
p3.show()
```

```
print"----------------------------------------"

print "Vector Plot with Normalized Length"
p4 = plot_vector_field((dS0/sqrt(dS0**2+dS2**2),dS2/sqrt(dS0**2+dS2**2)), (
    S0,0,paramDict[ST]), (S2,0,paramDict[ST]), color = 'blue', aspect_ratio
    =1, axes_labels=['$S_0$ Concentration','$S_2$ Concentration'], title = '
    Vector Plot of $(S_0, S_2)$')
p4 +=plot(paramDict[ST]-S0, (S0, startingx, paramDict[ST]), color = 'black'
    )
p4.show()

print"----------------------------------------"

print "END", datetime.datetime.now()
```

Appendix D

This appendix contains the Python code used to show A1 through A5 in Wang and Sontag's paper for the network in (1) [8].

```
import sympy as sp
import datetime
from sympy.abc import epsilon

print "START", datetime.datetime.now()

k1, k2, k2plusk3, k4, k5plusk6, k6, k7, l1, l2, l2plusl3, l4, l5plusl6, l6,
    l7, C1, C2, C3, C4, S0, S2, ET, FT, ST = sp.symbols('k1 k2 k2plusk3 k4
    k5plusk6 k6 k7 l1 l2 l2plusl3 l4 l5plusl6 l6 l7 C1 C2 C3 C4 S0 S2 ET FT
    ST')

S1 = ST- S0-S2-epsilon*C1-epsilon*C2-epsilon*C3-epsilon*C4
E = ET - C1 - C2
F = FT - C3 - C4

f = [l6*C4 - k1*S0*E+k2*C1, k6*C2-l1*S2*F+l2*C3]
g = [k1*S0*E - (k2plusk3+k7)*C1, k4*S1*E-(k5plusk6)*C2+k7*C1, l1*S2*F - (
    l2plusl3+l7)*C3, l4*S1*F-(l5plusl6)*C4+l7*C3]

g0 = [expr.subs({epsilon:0}) for expr in g]

C2fromC1 = sp.solve(g0[0], [C2])[0]
C4fromC3 = sp.solve(g0[2], [C4])[0]

reduced = [g0[1].subs({C2: C2fromC1, C4: C4fromC3}), g0[3].subs({C2:
    C2fromC1, C4: C4fromC3})]

#A2
C1sol = sp.simplify(sp.solve(reduced[0], C1)[0])
C2sol = sp.simplify(C2fromC1.subs({C1: C1sol}))
C3sol = sp.simplify(sp.solve(reduced[1], C3)[0])
C4sol = sp.simplify(C4fromC3.subs({C3: C3sol}))

#A4
J = sp.Matrix(g0).jacobian([C1, C2, C3, C4])
eigenJ = J.eigenvals()

#eigenJ.keys()[0] has - real part since S0+S2<=ST
#eigenJ.keys()[1] by above
#to check eigenJ[2] use code below to get lala >0
#     which implies - real part
#eigenJ.keys()[3] follows by symmetry

eigen3 = 2*eigenJ.keys()[2]
sqrtPart = eigen3.args[0]
otherPart = sum(eigen3.args[1:])
lala = sp.simplify(otherPart**2-sqrtPart**2)/4

#A3 follows from A4. See Hurwitz stable matrices
#A5 holds automatically
print "End at", datetime.datetime.now()
```

Appendix E

This appendix contains the Python code used to generate the plot in Figure 3.

```
import sympy as sp
import numpy as np
import datetime
from scipy import integrate
import matplotlib.pyplot as plt

print "START", datetime.datetime.now()

a1, a2, a3, a4, a5, a6, b1, c1, c2, d1, S0, S2, ET, FT, ST = sp.symbols('a1
     a2 a3 a4 a5 a6 b1 c1 c2 d1 S0 S2 ET FT ST')
S1 = ST - S0 - S2

#Our system
dS0dt = a1*S1*FT/(1+c2*S1+d1*S2) + a2*S2*FT/(1+c2*S1+d1*S2) - a3*S0*ET/(1+
    b1*S0+c1*S1)
dS2dt = a4*S1*ET/(1+b1*S0+c1*S1) + a5*S0*ET/(1+b1*S0+c1*S1) - a6*S2*FT/(1+
    c2*S1+d1*S2)

keys = [a1, a2, a3, a4, a5, a6, b1, c1, c2, d1, ET, FT, ST]

#Plug in parameter values
params = [5, 1, 6, 6, 8, 10, 1, 10, 3, 10, 3, 8, 8]
paramDict = dict(zip(keys, params))
dS0dtNum = dS0dt.subs(paramDict)
dS2dtNum = dS2dt.subs(paramDict)

def deriv(y, t):
    return [dS0dtNum.subs({S0: y[0], S2:y[1]}), \
            dS2dtNum.subs({S0: y[0], S2:y[1]})]

#set timescale to evaluate over and solve
time = np.arange(0, 10, 0.0001)
sol = integrate.odeint(deriv, [0, 8], time)

#Generate plots
labels = ["$S_0$", "$S_2$"]
for k in range(len(labels)):
    y = [sol[i][k] for i in range(len(sol))]
    plt.plot(time, y, label = labels[k])
plt.legend(loc = 1)
plt.title("$S_0, S_2$ with Respect to Time")
plt.xlabel("Time")
plt.ylabel("Concentration")
plt.savefig('output.png', dpi = 300)
plt.show()
plt.close()

plt.plot([sol[i][0] for i in range(len(sol))], [sol[i][1] for i in range(
    len(sol))])
plt.xlim([0, paramDict[ST]])
plt.ylim([0, paramDict[ST]])
plt.title("Trajectory of $S_0, S_2$ with Initial Value $(0, 8)$")
plt.xlabel("$S_0$")
plt.ylabel("$S_2$")
plt.savefig('output.png', dpi = 300)
plt.show()
```

```
plt.close()

print "END", datetime.datetime.now()
```