Bifurcation Analysis and Qualitative Optimization of Models in Molecular Cell Biology with Applications to the Circadian Clock

Emery D. Conrad

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
Mathematics

Robert Rogers, Chair
John Tyson, Advisor
Reinhard Laubenbacher
John Phillips
Michael Renardy

April 14, 2006
Blacksburg, Virginia

Keywords: bifurcation analysis, parameter optimization, circadian rhythms

Bifurcation Analysis and Qualitative Optimization of Models in Molecular Cell Biology with Applications to the Circadian Clock

Emery D. Conrad

Abstract

Circadian rhythms are the endogenous, roughly 24-hour rhythms that coordinate an organism's interaction with its cycling environment. The molecular mechanism underlying this physiological process is a cell-autonomous oscillator comprised of a complex regulatory network of interacting DNA, RNA and proteins that is surprisingly conserved across many different species. It is not a trivial task to understand how the positive and negative feedback loops interact to generate an oscillator capable of a) maintaining a 24-hour rhythm in constant conditions; b) entraining to external light and temperature signals; c) responding to pulses of light in a rather particular, predictable manner; and d) compensating itself so that the period is relatively constant over a large range of temperatures, even for mutations that affect the basal period of oscillation.

Mathematical modeling is a useful tool for dealing with such complexity, because it gives us an object that can be quickly probed and tested in lieu of the experiment or actual biological system. If we do a good job designing the model, it will help us to understand the biology better by predicting the outcome of future experiments. The difficulty lies in properly designing a model, a task that is made even more difficult by an acute lack of quantitative data. Thankfully, our qualitative understanding of a particular phenomenon, i.e. the observed physiology of the cell, can often be directly related to certain mathematical structures. Bifurcation analysis gives us a glimpse of these structures, and we can use these glimpses to build our models with greater confidence.

In this dissertation, I will discuss the particular problem of the circadian clock and describe a number of new methods and tools related to bifurcation analysis. These tools can effectively be applied during the modeling process to build detailed models of biological regulatory with greater ease.

Dedication

This dissertation is dedicated first and foremost to my wife Kaća. Her tolerance and determination to pick up the slack at home and with Sasha, our wonderful son, while I maniacally devoted myself to this research amazes me. I look forward to returning the favor, my love!

Also, to my parents, Andrew and Mary, and my sister, Heather, I owe a great debt of gratitude. The conversations of my youth, in which I mostly played but a fleeting role, nevertheless awoke my intellect; such a thing that cannot be put back to sleep! I am always eager to study and to discover. Thank you.

Acknowledgments

John (Tyson), I know being my mentor has not always been easy! Thank you for piquing my interest in computational biology and for *attempting* to show me what it means to be a scientist; I can only hope that I have indeed absorbed a piece of that wisdom. Thank you for your encouragement and support during the years of academia-versus-industry indecision! I am happy to say that I now whole-heartedly choose academia.

I also acknowledge the support of many others. Bob Rogers, early in my career you helped to pique my interest in dynamical systems; thank you. Thanks go to Chris Hong for collaborative work on temperature compensation (Chapter 5 is based on this collaboration with Chris and John) and your helpful advice related to the circadian clock. To Herbert Sauro, for your interest in Oscill8 and your most appreciated support of the tool within the community. A big round of applause and my gratitude go to Tongli Zhang for your insistent and tireless beating on Oscill8, uncovering more bugs in the early versions than I ever would have on my own. Keep up the good work, Tongli! Thanks are due Shengua Li; you also used the early versions of Oscill8. Thank you Kathy Chen for your willingness to help, whatever the cause, and of course for test driving Oscill8. Your good advice has made Oscill8 a better tool! (I look forward to more sage advice!!)

Contents

1 Introduction **1**
- 1.1 Circadian Rhythms . . . 1
- 1.2 Mathematical Modeling . . . 8
- 1.3 Existing Models of the Clock . . . 15
- 1.4 Oscill8 . . . 20
- 1.5 Summary . . . 21

2 Oscill8 **22**
- 2.1 Software Architecture . . . 24
  - 2.1.1 O8W, O8R and O8M Files . . . 25
  - 2.1.2 Data Integrity . . . 26
- 2.2 Philosophy Behind Interface . . . 27
- 2.3 Model Analysis with Oscill8 . . . 29
  - 2.3.1 Continuation . . . 31
  - 2.3.2 Parameter Twiddler . . . 34
  - 2.3.3 Period/Min/Max Sampling . . . 35
  - 2.3.4 Bifurcation Search . . . 36
- 2.4 Current Status, Limitations and Future Directions . . . 39

3 Bifurcation Matching **40**
- 3.1 Overview . . . 41
  - 3.1.1 A Simple Model . . . 43
  - 3.1.2 Compatible Structures: Problem T1 . . . 44
  - 3.1.3 Incompatible Structures: Problems T2 and T3 . . . 46
  - 3.1.4 Relative Fitness . . . 51
- 3.2 Algorithm . . . 52
  - 3.2.1 Multiplicative vs. Additive Perturbation . . . 53
  - 3.2.2 Adaptive Perturbation: σ and ρ . . . 54
  - 3.2.3 Full Algorithm . . . 56
- 3.3 Discussion of Results . . . 58
  - 3.3.1 Limitations and Future directions . . . 61

4 Circadian Models **66**
- 4.1 M0 . . . 66
- 4.2 M1 . . . 73
  - 4.2.1 Discussion of M1 Results . . . 78
- 4.3 M3 . . . 80
  - 4.3.1 Discussion of M3 Results . . . 82

4.4 M4 . . . 85
4.4.1 Discussion of M4 Results . . . 88
4.5 Summary . . . 93

5 Temperature Compensation **94**
5.1 Overview . . . 94
5.2 Algorithms for Tests I & II . . . 98
5.3 Discussion . . . 105

A ODE Files **108**
A.1 Bifurcation Matching . . . 108
A.1.1 Simple . . . 108
A.2 Circadian Models . . . 108
A.2.1 M0 . . . 109
A.2.2 M1 . . . 109
A.2.3 M3 . . . 110
A.2.4 M4 . . . 111
A.3 Temperature Compensation . . . 112
A.3.1 LG . . . 113
A.3.2 TH . . . 113
A.3.3 RS . . . 114

B O8R Files **115**
B.1 Optimization . . . 115
B.1.1 T1 . . . 115
B.1.2 T1-P . . . 116
B.1.3 T1-R . . . 117
B.1.4 T2 . . . 118
B.1.5 T2-R . . . 119
B.1.6 T3 . . . 121
B.1.7 T3-R . . . 123
B.2 Circadian Rhythm Models . . . 124
B.2.1 M0 . . . 124

List of Figures

1.1 Circadian Data 4
1.2 Cycling of *tws* and *wdb* Transcripts 6
1.3 Wiring Diagram of *D. melanogaster*'s Clock 9
1.4 Simulations vs. experimental data 11
1.5 Goldbeter's 1995 Model 16
1.6 Tyson et. al's 1999 Model 18

2.1 OSCILL8's interface 23
2.2 Print summary and plot properties 28
2.3 Batched One Parameter Continuation 31
2.4 One Parameter Continuation 32
2.5 Drag-and-drop combined two-parameter diagrams 33
2.6 RunSet Definition and Twiddler 34
2.7 Sample period/min/max results 35
2.8 Bifurcation search output 37

3.1 Compatible Structures S_0 and S_1 45
3.2 Optimization Results: T1 47
3.3 Incompatible Structures S_0 and S_2 48
3.4 Optimization Results: T2 49
3.5 Optimization Results: T3 50
3.6 Sample Generation Algorithm for Structure Optimization 53
3.7 Algorithm for ρ Adaptation 55
3.8 Main Algorithm for Structure Optimization 57
3.9 T1: 3D Fitness Landscape 58
3.10 T1: 3D Fitness Landscape 59
3.11 T2: 3D Fitness Lanscape 60
3.12 T3: 3D Fitness Landscape 61

4.1 M0 model 67
4.2 M0 model: HB at $n \approx 100$ 69
4.3 Optimized M0: Circadian Behavior 71
4.4 M1 model 73
4.5 M1 Initial 75
4.6 M1 Final TS and Data 77
4.7 M1 Bifurcation Structures 78
4.8 M3 model 80
4.9 M3 Final TS and Data 83
4.10 M4 model 85

4.11 M4 Final TS and Data 90
4.12 M4 Final Bifurcation Structure 91
4.13 M4 Light Affects 92

5.1 Test I Results 98
5.2 Test II Results 99
5.3 Sensitivity Rankings 100
5.4 Compensation Relation Check 101
5.5 Algorithm, Test I. 102
5.6 Oscillator Sensitivity 103
5.7 Algorithm, Test II. 104
5.8 The Resetting Hypothesis for RS 105

List of Tables

3.1 Main Control Parameters for Optimization 64
3.2 <match_diagram> and <match_point> Control Parameters 65
3.3 Algorithm and Perturbation Values and Meanings 65

4.1 M0 Parameter Values 68
4.2 M1 Parameter Values 76
4.3 M3 Parameter Values 82
4.4 M4 Parameter Values 89

5.1 Compensation relations 95
5.2 Basal Parameter Values: LG 96
5.3 Basal Parameter Values: TH 97
5.4 Basal Parameter Values: RS 97

Chapter 1

Introduction

1.1 Circadian Rhythms

The last few decades have seen an explosion in the study of circadian rhythms, the endogenous molecular clocks responsible for coordinating an organism's interaction with a cycling 24-hour environment. In the absence of any rhythmic environmental fluctuations, these endogenous clocks continue to drive rhythmic physiological processes and behaviors with an approximately 24-hour period ("circadian" literally translates to "about a day"). In addition, they are characterized by an ability to *entrain* to external light and temperature signals and are robustly *temperature compensated*, which means that they have the ability to adjust to changes in temperature with little or no change to the period of oscillation, even when perturbed by mutations to core clock components. These are key features in an organism's ability to adjust to changing environmental conditions.

Much has been discovered about the molecular mechanism of the circadian clock of many organisms, including *D. melanogaster* (the common fruit fly), and as it turns out, this mechanism shows a high degree of conservation between biological kingdoms. It is now understood that a complex network of a minimum of a dozen genes and their RNA and protein products comprise the core of *Drosophila*'s clock, and the mechanism includes both negative and positive feedback control. When we speak of the "clock," we should be careful to point out that there is no one, single clock. Many tissues in *Drosophila* express the molecular components of the clock and it appears that these components can play different

roles depending on the tissue. In *Drosophila* and other higher organisms, there is typically a *pacemaker* clock, or a main clock, which is responsible for driving peripheral oscillators. *Drosophila*'s pacemaker likely resides in a few clusters of lateral neurons (LNs) in the brain: the dorsal (LN_ds) and the ventral (LN_vs). Because molecular oscillations outside the brain do not seem to contribute to the circadian regulation of behavioral rhythms (Helfrich-Forster, 2002), we are primarily concerned with understanding studies of *Drosophila*'s brain tissue when we speak of "models" of the circadian clock in *D. melanogaster*. In mammals, there is an even clearer central pacemaker located in the *super-chiasmatic nucleus* (SCN), a small region of the brain that expresses the components necessary to drive an organism-wide time-keeping mechanism.

Molecular Components

In 1971, Konopka and Benzer discovered three rhythmic mutants of *Drosophila* mapping to a single locus on the X-chromosome, dubbed *per* (for *period*) due to the changed periodicity of *Drosophila*'s eclosion rhythm (Konopka and Benzer, 1971)[1]. Two of the mutations displayed altered periods of 19 (per^s for the short mutant) and 29 hours (per^l for the long mutant) with respect to the endogenous, 24-hour period. The third, per°, a null mutant, exhibited arrhythmic behavior (no overt rhythmicity at all). In the early 1990s, Hardin et al. suggested that the period protein (PER) feeds back to downregulate transcription of its own gene, producing a negative feedback loop (Hardin et al., 1990, 1992a,b).

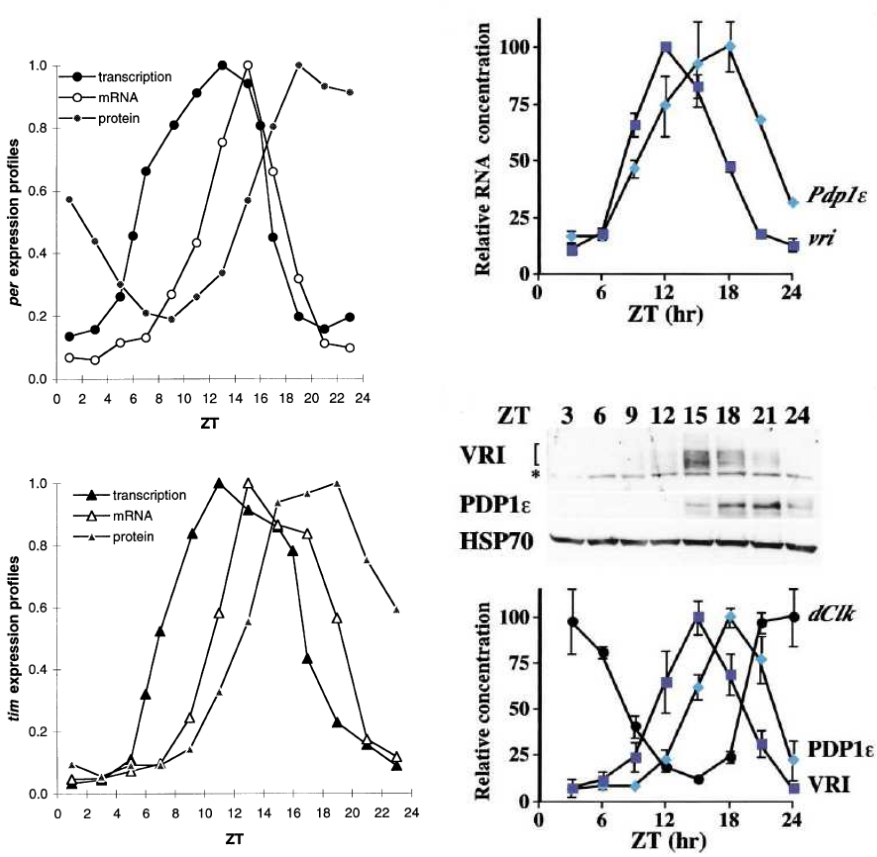
A few years later, in 1994, another gene *tim* (for *timeless*) was found to play a central role in the clock. In *tim* mutants, *per* mRNA showed no evidence of rhythmicity at all (Sehgal et al., 1994) and nuclear localization of the PER protein was found to be blocked (Vosshall et al., 1994), suggesting that *tim* mediates the negative feedback of PER on its own transcription. It turns out that PER forms heterodimers with TIM (Zeng et al., 1996) and at some point later in the cycle is translocated into the nucleus where PER (possibly PER-TIM) can inhibit transcription of both *per* and *tim*(along with a number of other genes). Whether or not the PER-TIM heterodimers can inhibit transcription is a matter of debate, though

[1]Eclosion refers to the emergence of the adult fly from the pupa.

it is clear that PER is capable of this repressive activity by itself, depending on its state of phosphorylation (Nawathean and Rosbash, 2004). What is also certain is that the formation of the PER-TIM heterodimer stabilizes the PER protein, blocking the negative effects of the phosphorylation activity by a casein kinase I-ϵ homolog, DBT (for doubletime) (Kloss et al., 1998; Price et al., 1998), and casein kinase II (Lin et al., 2002), which leads to its degradation via the proteosome pathway (Kloss et al., 1998; Grima et al., 2002; Ko et al., 2002; Naidoo et al., 1999; Akten et al., 2003), likely mediated by the function of other molecular components such as *slimb* (see below). In addition, protein phosphatase 2A, or PP2A, which acts directly on PER, is regulated in a circadian fashion (Sathyanarayanan et al., 2004). Both the formation of the heterodimer (and possibly homodimers of the PER protein) and the action of PP2A (independently) have the effect of introducing positive feedback into the system. In the first case, increasing amounts of PER, which is stabilized by binding TIM, predict less phosphorylation and subsequent degradation of PER. In the second case, PP2A works to stabilize PER and keep it in a less phosphorylated state (Sathyanarayanan et al., 2004). In *both* cases, because phosphorylation of PER potentiates its repressive activity, we have positive feedback. We are beginning to see the inklings of an interesting dynamical system.

The core mechanism for the negative feedback of PER and TIM on their own transcription relies on the transcription factors CLK and CYC ("clock" and "cycle", respectively), which form dimers and activate transcription of a number of genes, including *per* and *tim*. CLK-CYC is then rendered inactive by its association with phosphorylated PER (Rothenfluh et al., 2000b; Nawathean and Rosbash, 2004). This repression of transcriptional activity is quite possibly the result of degradation of CLK via DBT, and it may be that phosphorylated PER brings DBT in a complex to CLK-CYC in order to enact a transfer of DBT to CLK and CLK's consequent degradation (Hardin, 2005)[2]. In addition, the CLK-CYC dimer both inhibits and activates transcription of the *clk* gene via the intermediates *vri* and *pdp1*ε (described below), producing other positive and negative feedbacks in the system. Both *clock* mRNA and protein oscillate with a circadian period, while *cycle* is constitutively expressed. The CLK-CYC dimer is clearly a central component of the circadian clock machinery of *D. melanogaster*.

[2]Hardin refers to personal correspondence with W. Yu on this matter; these are unpublished results.



(a) So and Rosbash (1997) Fig. 2 (b) Cyran et al. (2003), Fig. 1

Figure 1.1: a) The basic rhythms related to the *per* and *tim* genes. The earliest curve to come up in both cases represents the transcriptional activity of each gene (as measured by a nuclear run-on assay), followed by the mRNA and protein concentrations over a roughly 24 hr. period. (Reproduced with permission from Macmillan Publishers Ltd. from So and Rosbash (1997)). b) The basic rhythms of *clk*, *vri* and *pdp1ε* mRNA and protein concentrations. (Reproduced with permission from Elsevier from Cyran et al. (2003)).

Here are a few details and other genes that are known to take a part in clock function:

- *slimb*, which produces an F-box/WD40-repeat protein (adapter for the SCF) which interacts preferentially with phosphorylated PER and stimulates its degradation (Grima et al., 2002; Ko et al., 2002). This may be the means by which PER is targeted for degradation by the proteosome once it has been phosphorylated by DBT.

- *casein kinase IIα*, or CKIIα. The CKII kinase affects the stability of PER in conjunction with DBT, though it is not exactly clear how the role of DBT differs from CKIIα (Lin et al., 2002; Nawathean and Rosbash, 2004).

- *twins* (*tws*) and *widerborst* (*wdb*), are candidate adapter proteins which make PP2A

specific to PER and/or TIM. *tws* will play a role in our proposed models below.

- *shaggy*, or *sgg*, whose protein product is responsible for phosphorylating TIM, which appears to affect the ability of the inhibition complex including PER and TIM to enter the nucleus[3]. SGG is constitutively expressed (Martinek et al., 2001).

- *vrille*, or *vri*, codes for a transcription factor which represses *clk* transcription and is activated by the CLK-CYC dimer (Cyran et al., 2003; Glossop et al., 2003). See Figure 1.1(b) for data on *vri* and *pdp1ε*.

- *Par Domain Protein 1*, or *pdp1ε*, codes for a transcriptional factor which activates *clk* transcription and is activated by CLK-CYC (Cyran et al., 2003). Notice the antagonistic roles of *vri* and *pdp1ε*. See Figure 1.1(b) for data on *vri* and *pdp1ε*.

- *cryptochrome*, or *cry*, codes for a blue-light photoreceptor involved in light input to the clock. Our models will utilize *cry* for light input (see below).

- *pigment dispersing factor*, or *pdf*, an output of the clock expressed in the LN_vs. This gene will not be part of the proposed models.

Now let us look at the several of these extra components a bit more closely since they are going to play significant roles in the ensuing discussion.

Protein Phosphatase 2A

Protein phosphatase 2A (PP2A) is actually a family of phosphatases, comprised of a catalytic subunit, C, along with a regulatory subunit, A. These subunits together form the scaffolding onto which a variable regulatory subunit, B, can attach (Janssens and Goris, 2001). In the case of *Drosophila*, the catalytic subunit is *mutagenic star*, or *mts*, and there are four genes encoding for the B-subunit, two of which are involved in the circadian clock (Sathyanarayanan et al., 2004); they are *tws* and *wdb*. This phosphatase appears to act directly on

[3]Overexpression of *sgg* results in advanced nuclear entry of the inhibitory complex. See (Martinek et al., 2001) for details.

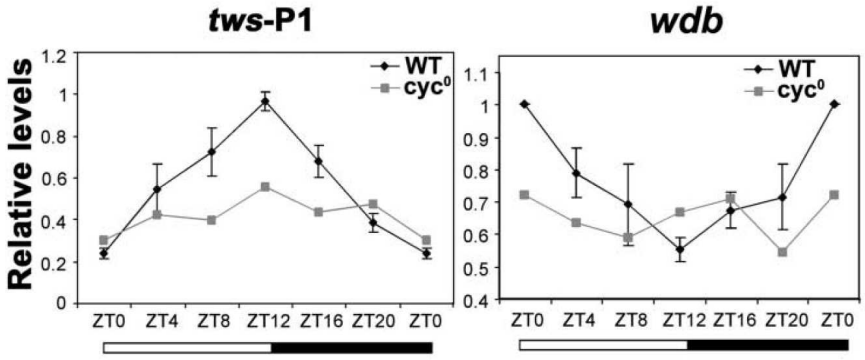


Figure 1.2: Cycling of *tws* and *wdb* mRNA transcripts. (Reproduced with permission from Elsevier from Sathyanarayanan et al. (2004)).

PER and counteracts the effect of DBT and/or CKII (Sathyanarayanan et al., 2004). Interestingly, PP2A is regulated in a circadian fashion, which is in contrast to the constitutive kinase activity on PER.

Levels of *tws* (two transcripts) and *wdb* mRNA cycle under circadian clock control (i.e., they cycle except in the cyc^0 mutant), although they are out of phase with one another, indicating different roles in the clock in connection with phosphatase activity (see Figure 1.2). They also surprisingly have *opposing* effects on behavioral rhythms (overexpression of *wdb* results in period lengthening, whereas overexpression of *tws* results in shorter period for several days, followed by arrhythmia). *tws* appears to be more strongly expressed, exhibiting a 5-fold cycling (of mRNA), whereas *wdb* mRNA has only 2-fold cycling. In addition, *tws* has a CACGTG E-box in the first intron of its P1 transcript making it a direct target of CLK-CYC. Overexpression of *wdb* affects the rhythm by increasing the oscillator period and increasing levels of PER while reducing the amplitude of PER oscillation from 5-fold to 2-fold. This overexpression also advances PER nuclear entry, likely because of higher levels. Sathyanarayanan et al. (2004) conclude that

- PP2A rhythmically dephosphorylates and stabilizes PER at specific times in the circadian cycle and sustains PER oscillations through mechanisms that do not depend on the cycling of *per* mRNA, and
- PP2A normally serves to promote PER stability and also most likely its nuclear localization. (The effect on nuclear localization is consistent with (Vielhaber et al., 2000) in the mammalian system.)

and further predict that

- the balance between the mechanisms that stabilize PER (dephosphorylation) and those that destabilize it (phosphorylation) is *critical* for maintaining the cycling of the PER protein, and
- under conditions where *per* mRNA does not cycle, PER protein cycling is maintained by the rhythmic expression of *wdb* and *tws*.

These details will be explored in the models constructed below.

Light Input Pathway

One of the most interesting features of the circadian clock is its ability to adapt to changing conditions. In particular, the ability of light to entrain or reset the clock. It is this ability that allows our bodies to eventually adjust to a different timezone after an overseas flight (although sometimes an uncomfortable period of phase shift—what we experience as jetlag—must be endured).

One of the key photoreceptor genes in *D. melanogaster* is *cryptochrome*, which belongs to a family of blue light-responsive flavoproteins, some of which function as photolyases (although *cry* does not). In wild-type flies, *cry* RNA is highly expressed in the LNs and cycles with a circadian period (Emery et al., 1998), while CRY protein accumulates in the absence of light (but cycles under normal light conditions). This indicates that in the LNs, CRY is light controlled rather than clock controlled. In peripheral tissues, CRY may, in fact, be a core clock component (Ashmore and Sehgal, 2003).

In 1996, it was observed that a pulse of light causes rapid degradation of TIM (Myers et al., 1996) and had the overall effect of resetting the phase of the clock. In a cryptochrome mutant, cry^b, a missense mutation at a flavin-binding residue that results in barely detectable levels of CRY (see (Stanewsky et al., 1998), Fig 4 A & D), it was found that TIM levels did *not* crash in response to light and remained at high levels throughout the day. This is explained by the fact that CRY^b is both insensitive to light and unstable (Stanewsky et al., 1998; Lin et al., 2001; Froy et al., 2002; Ashmore and Sehgal, 2003).

The current hypothesis is that CRY changes its conformation in response to light (Lin et al., 2001), binds to the clock proteins (i.e., TIM), and transmits a signal that leads to the degradation of itself and of TIM, although the relative sequence of these events is not yet clear (Ashmore and Sehgal, 2003). Although Cashmore (2003) specifically states that the binding of TIM and CRY does not appear to be light-dependent, there seems to be mounting evidence that indeed the binding *is* light dependent (Busza et al., 2004; Dissel et al., 2004). In our model M4, we will assume that this is the case.

1.2 Mathematical Modeling

Our ultimate goal is to understand how the molecular details of the clock combine to give us the physiology observed. To do so, we take a step back and try to create a coherent, unified picture that represents *all* the details described above. At such a high level, it is often useful to sketch a picture or cartoon such as that seen in Figure 1.3. In order to make any sense of such a picture, we need to translate it into an object that can be rigorously studied. Although there are many choices for such a formulation, to keep things as simple as possible in the beginning we choose nonlinear ordinary differential equations (ODEs), which relates the rate of growth or decay of the various biochemical species of a system to rate laws describing the interactions between the same species. For a detailed description of the basics of modeling regulatory networks with ODEs, see Conrad and Tyson (2006).

More precisely, a system of nonlinear ODEs is defined as

$$\dot{x} = f(x, p), \qquad x \in \mathbb{R}^n, p \in \mathbb{R}^m \tag{1.1}$$

where x is an n-dimensional state vector whose components represent the concentrations of the system's biomolecular species, p is an m-dimensional parameter vector whose components represent the kinetic constants and parameters, and $f(x, p) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the collection of rate laws that govern the evolution of the state of the system, x. For example, each species

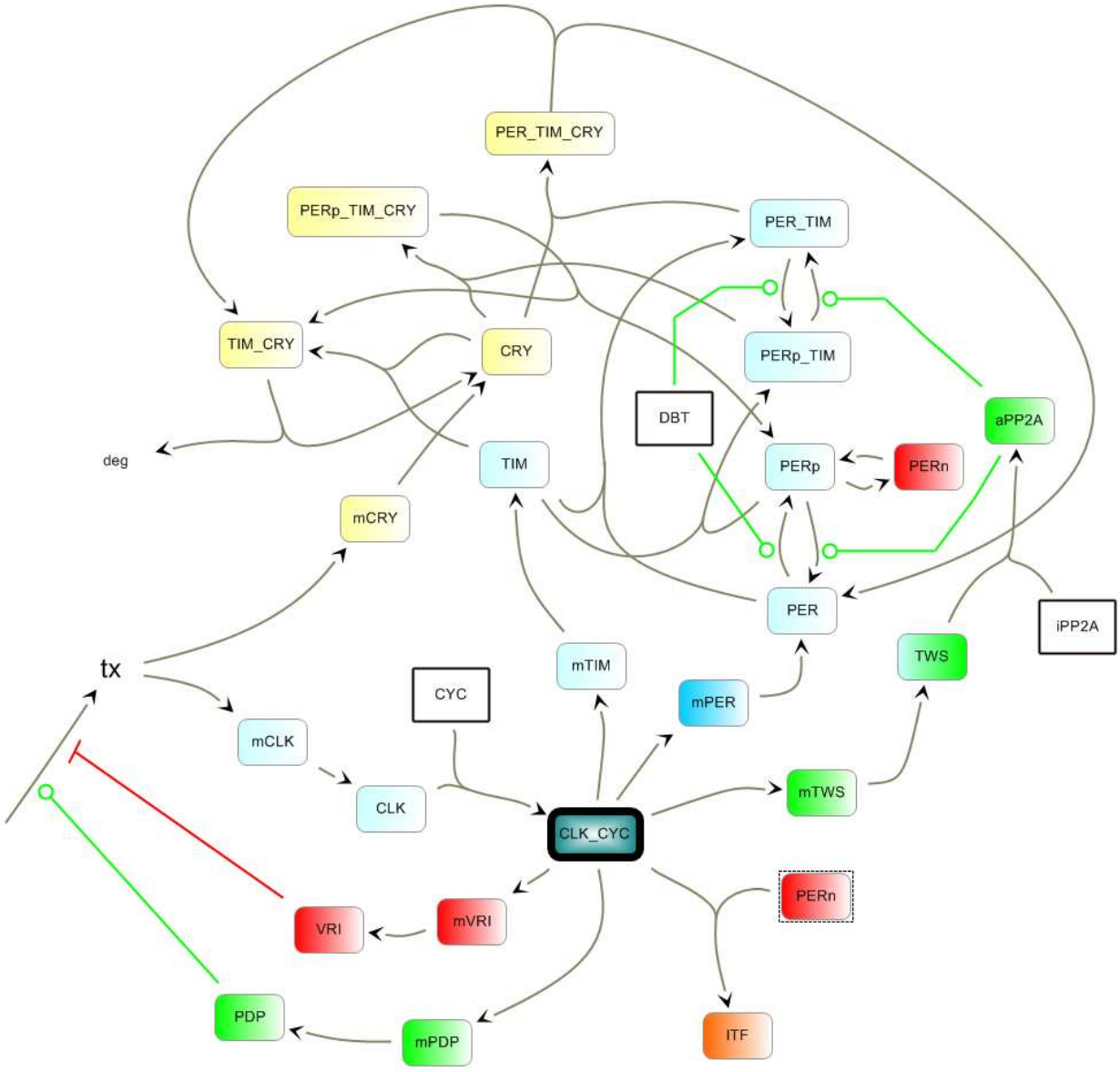


Figure 1.3: Sketch or wiring diagram representing a particular hypothesis for the molecular clock mechanism of *D. melanogaster*. Although this sketch resembles a monkey's head, it is in fact model M4, and will be analyzed in Section 4.4.

x_i has an equation such as

$$\frac{dx_i}{dt} = \text{synthesis} - \text{degradation} - \text{binding} + \text{release} + \ldots$$

describing all the processes that contribute to the growth and decay of x_i. Typically, n and m are moderately large (10 or more), and f is rational and smooth, i.e., has continuous derivatives up to several orders. (Unless explicitly stated, it will be assumed throughout this dissertation that $f(x, p)$ is sufficiently differentiable up to several orders). The choice of ODEs requires a few simplifying assumptions.

1. The system is a "well-stirred" chemical reactor (i.e., component concentrations are space-independent). This is a reasonable approximation for many systems, including the cell cycle and circadian rhythms because the time scale of the phenomenon under study is much slower than the diffusion rate of proteins in the cell. For reference, given the typical diameter of a cell to be 10^{-3} cm and a typical diffusion constant for a protein in aqueous solution to be $D = 10^{-7}$ cm^2/s, we can calculate the average time for a protein to diffuse across a cell to be $t = \frac{(10^{-3}\text{cm})^2}{2\times 10^{-7}\text{cm}^2/\text{s}} = 5s$. If diffusion is 10-fold slower in cytoplasm, then the average time to cross a cell is roughly 1 min. If we are studying a phenomenon whose time scale is closer to 1 min, then we need to include spatial dimensions into a partial differential equation (PDE) model.

2. The concentrations, x, are continuous functions of time. A spherical cell of diameter 10^{-5} m has a volume of roughly 0.5×10^{-15} m^3 $= 5 \times 10^{-13}$ L. Given a typical concentration of a specific regulatory protein to be 10 nM, we calculate $10^{-8}\ \frac{\text{mol}}{\text{L}} \times 6 \times 10^{23}\ \frac{\text{molecules}}{\text{mol}} \times 5 \times 10^{-13}\ \frac{\text{L}}{\text{cell}} = 3000\ \frac{\text{molecules}}{\text{cell}}$. For a reaction volume containing 3000 molecules, we are justified in using ordinary differential equations to describe changes in a continuous variable $x_i(t)$. Were the concentration drops below 1 nM, we would need to reformulate the model in terms of stochastic variables to capture the effects of molecular noise in the dynamical system.

In any case, ODEs provide a good starting point since there is a great deal of theory related to ODEs. Other, more sophisticated modeling techniques such as PDE or stochastic analysis

build on the theory developed for ODEs.

Simulations

Now, the first and most natural step in analyzing a mathematical model is to run simulations given a plausible initial state. Generally speaking, there exists a solution to any well-posed initial value problem (due to existence and uniqueness). But practically speaking, we cannot find it in closed form. We therefore typically rely on numerical approximation to the solution using a stiff ODE solver. However, for a moment, suppose that we do have a closed-form solution to an ODE model of the circadian clock (for just the concentration of PER, P) that looks like $P(t) = A + B\sin(\omega t) + C\cos(\omega t)$. Since the circadian clock has a period of roughly 24, $\omega \approx \frac{2\pi}{24}$ and we can simulate the behavior of the model for different values of A, B and C and compare the results with experimental data. Figure 1.4 shows the typical situation.

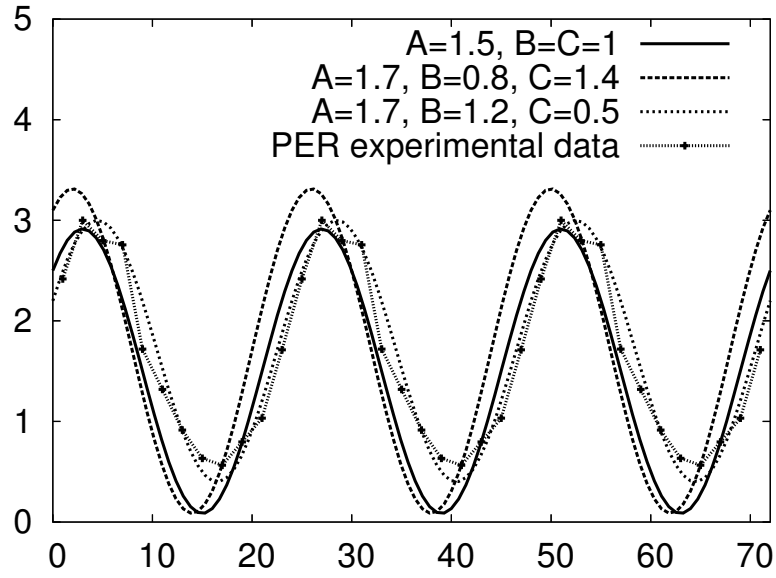


Figure 1.4: Simulations of $P(t) = A + B\sin(\omega t) + C\cos(\omega t)$, with $\omega = \pi/12$. Here we have three different model simulations overlaid on actual experimental data for levels of the PER protein (taken from Figure 1.1(a), levels scaled to 3 instead of 1 for convenience).

We have plotted three different curves of our model equation $P(t)$ along with experimental data (scaled for convenience). As we can readily see, one of the model curves "fits" the experimental data very well. In fact, if all we wanted to model was the PER data plotted in this graph, our model equation (for those particular values of A, B and C) would be pretty

good. (On the other hand, the "model" given here has no connection to the molecular details of the clock, and is therefore of rather limited use.)

The next logical question to ask is: how do we choose good values for A, B and C so that, at the very least, we have confidence that our model reproduces experimental data? Colloquially, how do we "fit" our model to experimental data? Was my choice of A, B and C just lucky in Figure 1.4? The answer, of course, depends upon the question we hope the model can help us to answer. Often, we ask quantitative questions such as "at what time will the level of protein X drop below 2.75?" or "what factor can I multiply parameter p_1 by to get a 10% increase in the period of oscillation?" Other times, we are more interested in qualitative questions such as "will flies become arrhythmic if I double component X at time $t_0 = 1, 5$ or 6?" or "which reactions most heavily influence the period of oscillation?" These questions are all relevant, but need to be addressed separately.

Quantitative Parameter Optimization for Time Series Data

One of the main issues that modelers of complex biological systems have to deal with is accounting for all the details represented by an increasingly rich data set. Modern molecular biology is capable of producing a great deal of information that helps to understand the composition and quality of the interactions that comprise these networks. Beyond sifting through this data to obtain a realistic picture of the network topology, we still have the problem that the data obtained is often much more useful in uncovering these interactions than giving quantitative estimates on rate constants. So, as modelers, one of our main tasks is to choose a good parameter set—one that makes the model not only explain the known data but also gives it as much predictive value as possible.

The traditional attack of this problem requires modelers to combine their wits with a healthy dose of dynamical systems theory, tweaking parameters a bit at a time attempting to adjust the behavior to match experimental results. As mentioned earlier, analytical results are largely impossible given the form and degree of the equations involved in the right hand sides of the system of ODEs, so we must rely on numerical tools. There are many tools, both for the study of dynamical systems and bifurcation analysis (see the next sections) that

continue to be the basis for any numerical attack on models in biology. Unfortunately, many of these tools have severe limitations in the context of mathematical biology (discussed is Chapter 2).

Fortunately, there are automated parameter estimation schemes that use sophisticated numerical algorithms to "fit" parameters to experimental data (see for example Zwolak et al. (2005) and Jason Zwolak's parameter estimation software[4]). At the very least, such methods have the effect of reducing the region of interest in our high-dimensional parameter space by producing rough estimates and can help to tell us whether or not our model is even capable of achieving realistic results (i.e., when no good parameter "fits" are found, we might be able to throw out candidate models). This type of automated estimation can therefore help us to evaluate our basic understanding of the process we are modeling (i.e., a critique of the wiring diagram) as well as provide some useful reduction of our valid parameter space.

Unfortunately, because typical automated parameter estimation is based on time series data, it provides little dynamical information about our system. It simply tells us that given a certain set of parameters we can mimic the behavior of the system to some degree of accuracy. Making valid predictions to help direct experiments would typically require knowledge of dynamical possibilities, requiring results from bifurcation theory (see below). Other tools, such as those mentioned above, must be used to get more detail once a good set of parameters is found. This thesis will partially focus on providing new methods of attacking such problems, both by providing new tools and investigating new theoretical techniques of optimization which rely on bifurcation information.

Qualitative Optimization and Dynamical Systems Theory

Optimization, as just discussed, attempts to fit a model to either experimental data or to conceptual constraints related to its dynamical properties. When we are primarily interested in qualitative features of dynamical behavior, these dynamical constraints are derived from the *bifurcation structure*, a concept we will develop later in this thesis (see Chapter 3). It turns out a modeler can make a great deal of headway in model development and fitting

[4] `http://mpf.biol.vt.edu` under the "Parameter Estimation" link.

with just small hypotheses related to bifurcation information.

As just a quick introduction to the salient points of bifurcation analysis, we recall that generally speaking, it is the study of how the behavior of a dynamical system depends on its parameter values. Most important to our analysis are *Hopf*, *saddle-node*, and *saddle-node on invariant circles* bifurcations (denoted HB, SN, SNIC, respectively).

To understand these bifurcations, suppose that

- there is a smooth curve $p = v(\alpha) : \mathbb{R} \mapsto \mathbb{R}^m$ through parameter space such that $v' \neq 0$,
- $\hat{x}(\alpha)$ is a steady state solution of (1.1); that is, $f(\hat{x}(\alpha), p(\alpha)) = 0$, so that we think of the steady state solution as a function of α,
- $A = \frac{\partial f}{\partial x}(\hat{x}(\alpha))$ is the Jacobian matrix evaluated at the steady state,
- $\lambda_i(\alpha)$, where $i = 1 \ldots n$, are the eigenvalues of A.

If Re $\lambda_i(\alpha) < 0$ for all i, then $\hat{x}(\alpha)$ is stable, meaning that nearby solutions tend toward $\hat{x}(\alpha)$; otherwise, it is unstable (meaning that there are nearby solutions that *do not* tend toward $\hat{x}(\alpha)$). We say that a *local bifurcation* occurs in system (1.1) at $p^* = v(0)$ as α passes through zero if for some eigenvalue, λ_k,

$$\text{Re } \lambda_k(0) = 0. \tag{1.2}$$

As just mentioned, in biological systems, the bifurcations of primary importance are

- saddle-node (SN), generating the possibility of bistable behavior as a result of positive feedback (bistability is actually typically characterized by a pair of SN bifurcations). A saddle-node bifurcation is characterized by the collision, and subsequent disappearance, of two steady state solutions as α crosses zero. At $\alpha = 0$, these two steady states become one and in fact

  $$\lambda(0) = 0$$

  for some eigenvalue λ. So on one side of $\alpha = 0$, there are locally two steady states and on the other side, none (locally). Saddle-node bifurcations are a classic way of producing bistability in biochemical systems and are the hallmark of positive feedback.

- Hopf (HB), generating oscillation as a result of negative feedback. A Hopf bifurcation, on the other hand, is characterized by a change in stability of a steady state as we move along v together with the birth of a one-parameter family of small amplitude limit cycle solutions in the vicinity of $\alpha = 0$ (parameterized by α). In this case, a complex conjugate pair of eigenvalues satisfy the additional conditions (along with eq. (1.2))

$$\frac{\partial \text{Re } \lambda}{\partial \alpha}(\alpha = 0) \neq 0, \tag{1.3}$$

$$\text{Im } \lambda(0) \neq 0 \tag{1.4}$$

  Hopf bifurcations are a classic way of generating spontaneous oscillation in biochemical systems and are the hallmark of negative feedback.

- saddle-node on invariant circles (SNIC), an infinite period (global) bifurcation that occurs as a closed, isolated, periodic solution of (1.1) (a limit cycle) collides with a saddle-node. SNIC bifurcations are the result of an interesting interaction of a saddle-node and Hopf bifurcation. A SNIC occurs as α crosses through zero if the limit cycle solutions thrown off from a Hopf bifurcation (not at $\alpha = 0$) collide with a saddle-node bifurcation at $\alpha = 0$ (hence a saddle-node of *invariant circles*, a.k.a. limit cycles). In this case, the period of the limit cycle oscillation goes to infinity as $\alpha \to 0$.

A more comprehensive treatment of the mathematical theory of bifurcation analysis can be found in Kuznetsov (2003), but this is a sufficient introduction for the purposes of this research. We will primarily be interested in the consequences of these bifurcations related to biology and harnessing the mathematical techniques available.

1.3 Existing Models of the Clock

We concentrate here on *molecular* models of the rhythm as opposed to models that attempt to fit a known oscillatory model to circadian data (see, for example, Gundel and Spencer (1999)).

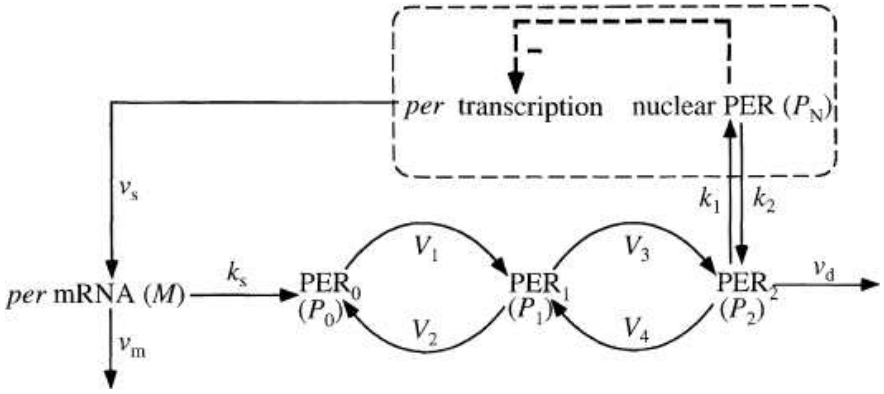


Figure 1.5: Goldbeter's 1995 model based on *per* and bisphosphorylation of PER prior to nuclear localization. There are a total of 5 state variables—mRNA, protein, protein plus one (and two) phosphate group(s), and a nuclear form of the bisphosphorylated protein—and 18 parameters governing the behavior of this model. (Reproduced with permission from the Royal Society of London from Goldbeter (1995)).

Leloup & Goldbeter: Negative Feedback

Several molecular models of *D. melanogaster*'s clock have appeared over the years, beginning with Albert Goldbeter's first model in 1995 (Goldbeter, 1995). At that time, only *per* had been identified as a major component of the clock. This model incorporates the *per* mRNA and protein as well as requiring a two-level phosphorylation step prior to PER's entry into the nucleus, where it inhibits the transcription of its own mRNA (see Figure 1.5). This simple model mimics the basic Goodwin model (Goodwin, 1965) of feedback inhibition. Goldbeter shows how changing the rate of PER degradation in the model changes the period of oscillation by doing some simple bifurcation analysis. This is the type of analysis that can uncover and characterize the possible modes of behavior for such models. In this particular case, it is also biologically relevant to discuss such changes since mutations to the *per* gene could very plausibly change the rate of degradation of the protein and could be used to explain various modes of behavior for mutant strains of the fly[5].

Again in 1998, Leloup and Goldbeter (1998) proposed a revised version of the 1995 model that includes *tim* and its byproducts. The model retains the bisphosphorylation step (now on both PER and TIM) but adds the formation of the PER-TIM complex (of the bisphosphorylated forms) prior to nuclear entry. This nuclear complex subsequently becomes the inhibitory element of the model (see Figure 1 of Leloup and Goldbeter (1998)).

[5]Goldbeter plots the period of oscillation vs. the maximum rate of degradation of PER, marking where the various *per* mutants would lie along the curve.

Of particular note in this model is the role placed on light input. At the time this model was proposed, it had been determined that TIM was rapidly degraded upon the organism's exposure to light. Leloup and Goldbeter introduced a varying rate of TIM degradation to explain this phenomenon. Their results discuss the model's response to light, which can be succinctly characterized by a *phase response curve* or *PRC*, a standard plot depicting the effect of a stimulus on an oscillator. PRCs have been determined experimentally for *Drosophila*, so it is fitting to compare a model's version of these plots to experimental data. The results presented in Leloup and Goldbeter (1998) are consistent with experimental results, given certain assumptions about how light affects the degradation kinetics of TIM. (They found good results for a two-fold increase in TIM's degradation rate constant for a three hour period. They also did a systematic study of changing strengths and durations of this kinetic constant, which gives a spectrum of PRCs showing various behavior.)

It is also interesting to point out that this model is capable of much more complex dynamical behavior than the simpler model of 1995. Indeed, Leloup and Goldbeter published several subsequent studies of this model (Leloup and Goldbeter, 1999; Leloup et al., 1999; Gonze et al., 2000; Leloup and Goldbeter, 2000, 2001)), some of which detail this model's ability to generate both chaos and birhythmicity. While these facts are interesting from a mathematical point of view, it is not clear whether they all play a biological role in *Drosophila*. In addition, Leloup and Goldbeter have published a paper describing the model's ability to explain the rather strange observation that the circadian clock can be abolished (for some time) and restored via light pulses (Leloup and Goldbeter, 2001).

The models of Leloup and Goldbeter rely on a central negative feedback mechanism that these proteins exert on their own transcription. Other molecular models have been proposed that rely on an additional positive feedback to generate oscillation, which points out that there is more than one way to skin this particular cat!

Alternate Models: Positive Feedback

In 1999, Tyson & Hong proposed a model of the circadian clock in *Drosophila*, emphasizing the role of dimerization and proteolysis of PER and TIM (Tyson et al., 1999). Central to

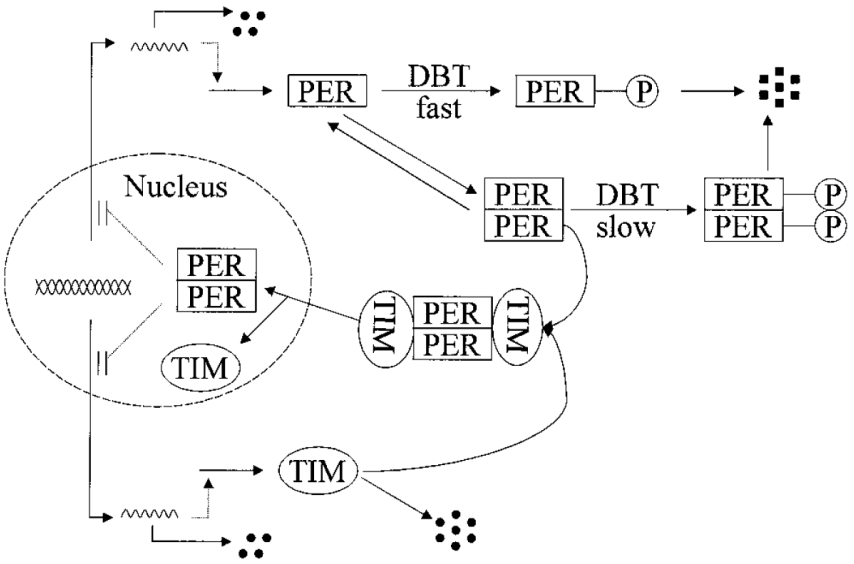


Figure 1.6: Tyson et. al.'s model incorporating positive feedback via an autocatalytic mechanism of PER. The basic similarity to the models of Leloup & Goldbeter is clear, but the addition of the homodimer and rapid degradation of the monomeric form of PER sets up a completely different dynamic in the system. (Reproduced with permission from the Biophysical Society from Tyson et al. (1999)).

this notion is the kinase DBT, which has been described above. This model relies on fast phosphorylation, and subsequent degradation, of PER monomers via DBT and the relative stability of PER-PER homodimers, which sets up an autocatalytic or positive feedback mechanism for PER.

This effect is indirect, but clear: the more PER present in the system, the more stable homodimers, which results in more total PER remaining in the system. The introduction of this positive feedback creates the possibility of *switch* or *hysteretic* behavior, a feature not generally present in systems based on negative feedback alone. *Hysteresis* occurs when there is a range of parameter values (or a region of parameter space, more generally) where there are multiple stable steady states. It is characterized by parameter "thresholds" at which the system jumps from one stable steady state to another. Crossing these thresholds is often an irreversible process, as the thresholds can be directional.

Smolen et. al.

Other models with limited molecular detail and time-delay include those of Smolen et. al. (Smolen et al., 2001, 2002, 2004). One of these models is very similar to those just dis-

cussed and the other incorporates *distributed* time-delay in addition to considering a large number of phosphorylations of PER prior to its degradation. Both include the CLK-CYC method of indirect inhibition on the transcription of *mPer*. The 2004 model includes the effects of *vri* and *pdp1ε* into the model, and is certainly interesting. The main objection would be that explicit delay has been used to cover up missing details, although we have no exact proposal of how to remedy this. Their model does not suffer from the same failing as model M4, as you will see in Section 4.4.

Other Models

There are a handful of other models which attempt to illuminate certain aspects of the circadian clock, including the discrete time-delay models proposed by Olde Scheper et. al. (Scheper et al., 1999b,a) and Lema et. al. (Lema et al., 2000).

Olde Scheper et. al. proposes the very simple model

$$\begin{aligned}\frac{dM}{dt} &= \frac{r_M}{k+P^n} - q_M M \\ \frac{dP}{dt} &= r_P M(t-\delta)^m - q_P P\end{aligned}$$

which is a form of Goodwin's negative feedback oscillator, with the time delay made explicit (as opposed to adding a number of molecular components in a chain of reactions). The first term in the dM/dt equation is the expression of the protein's inhibitory effect on transcription, while the first term of the dP/dt equation introduces a time-lagged dependency of the protein's synthesis on its mRNA. The second term of each equation expresses a background degradation. In the context of this model, light input is discussed as a variation of the protein degradation constant, q_P, much like in Leloup and Goldbeter's 1998 model. Likewise, the model of Lema et. al. (Lema et al., 2000) uses only a single equation, lumping both the mRNA and protein into a single species and incorporating both the nonlinear inhibition and time-lag into a single term. This is done in the name of preserving only the smallest amount of information necessary to sustain an oscillation and then asking a number of questions about the sensitivity of the oscillation with respect to the parameters, in addition to

computing light-induced PRCs.

1.4 Oscill8

As the amount of data available increases and we understand more about the intricate details of the system we are trying to model, we require more and more sophisticated models to account for these details. This has the effect of increasing the number of state variables and parameters and makes our analysis of any corresponding model more difficult. As described above, there are more than a dozen genes whose mRNA and protein products all interact with each other to produce the circadian clock mechanism. Any realistic model of this mechanism that hopes to have a chance of answering detailed questions would require possibly hundreds of state variables and many more parameters. Given the current set of tools, this type of analysis is grueling and quite difficult.

One of the aims of my research has been to enhance the tools related to bifurcation analysis and parameter optimization so that modelers can worry less about the mathematical and algorithmic details of their analysis and worry more about the biology of the system and how it is represented as a model. *Oscill8* gives a modeler a simple, user-friendly environment to analyze ODE models by providing services related to simulation, bifurcation analysis and parameter optimization. It allows the user to interact directly with graphic data generated during an analysis and also acts as a sort of modeler's notebook, associating text with diagrams and storing a history of what parameter values and numerical algorithms were used to generate data. This tool will be discussed in detail in Chapter 2 and will be used extensively and exclusively in this dissertation.

1.5 Summary

To summarize, in this dissertation you will find the following:

- A discussion of the rationale for and the development of OSCILL8, a tool that incorporates many third party tools and which is capable of greatly enhancing the day-to-day work of a computational biologist interested in modeling complex regulatory networks with non-linear ODEs (see Chapter 2). We will also include a tutorial section highlighting the most useful features of OSCILL8.

- A new parameter optimization technique based on optimizing bifurcation structure (see Chapter 3). This idea will be used heavily in Chapter 4 to achieve rapid construction and fitting of complex models of the circadian clock.

- An ensemble of new models of *D. melanogaster*'s circadian clock incorporating a few novel features of interest to the biological community. This set of models will be used to show how a modeler can quickly increase the level of complexity of a model without fear of getting completely lost in parameter space. OSCILL8 will play a central role in this material. (See Chapter 4.)

- A discussion of new techniques developed to compare different models' ability to compensate circadian period for changes in temperature (see Chapter 5).

Chapter 2

Oscill8

Oscill8 was specifically designed to overcome many of the challenges that exist in the current crop of dynamical systems tools. In particular, the difficulty of bifurcation analysis (which is, in itself, a complex mathematical task involving a deep understanding of dynamical systems) has been compounded by the fact that the tools that exist were primarily developed to investigate and research the phenomena of bifurcations, and not designed to be usable and friendly at a higher level, where we might be interested in the consequences of a particular bifurcation structure (see Chapter 3 for a discussion of bifurcation structure). Some tools, such as AUTO (Doedel et al., 1997) and CANDYS/QA (Jansen, http://www.agnld.uni-potsdam.de/w̃olfgang/ca-ov.html), require the user to write C/C++ or Fortran code and compile it prior to performing any analysis. CONTENT (Kuznetsov, 1998) will automate the compilation step for you, but still requires you to formulate a C compatible version of the model, whereas XPPAUT (Ermentrout, 2002) has its own interpreter (does not require compilation) and MATCONT (Dhooge et al., 2003) relies on Matlab's symbolic capabilities to define the model and derivatives of the right hand side of the system of ODEs.

Regardless, the start-up issue is not the main problem for the type of analysis relevant to computational biology. What does a computational biologist need? Besides a seamless method for creating models and automatically generating differential equations from them (for which we use JDesigner, http://jdesigner.org, also a graphic layout tool that helps produce nice wiring diagrams, e.g., as in this dissertation), they need access to higher-level bifurcation analytic information derived from their model that is seamlessly integrated

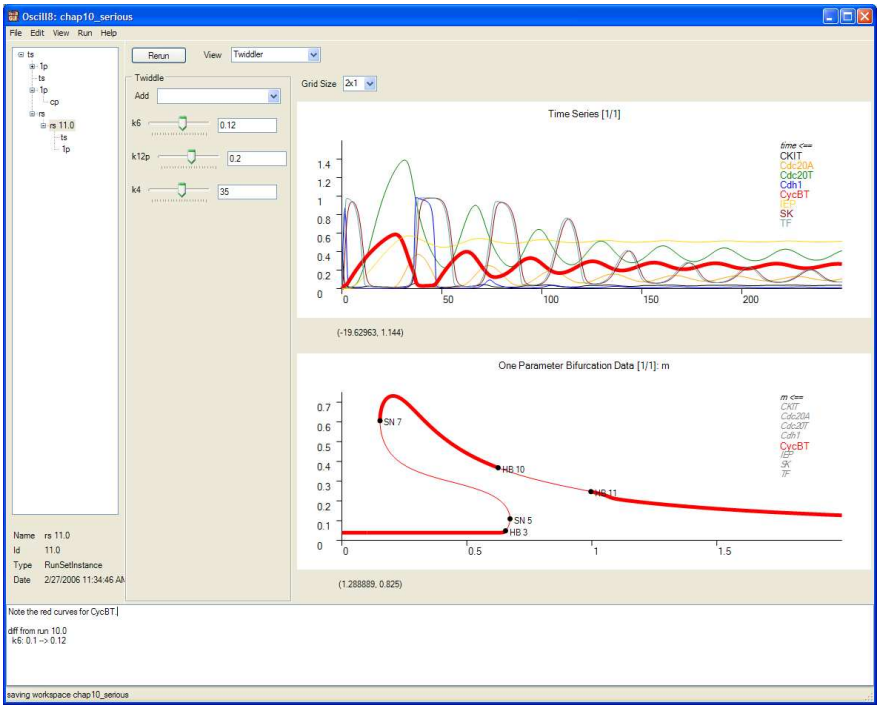


Figure 2.1: The interface to OSCILL8 includes a) a sidebar at left with a hierarchical listing of generated diagrams, b) a main graphics area for display of data, c) an area at the bottom for notes related to diagrams, and d) an optional "twiddler" area that allows for adjusting parameters. OSCILL8 keeps track of all initial conditions, parameter values and other meta-data associated with state variables and parameters needed to regenerate each graphic (via the "Model Data" interface).

into a easy to use graphical user interface (see Figure 2.1). Functionality such as batched one- and two-parameter continuation with interactive graphic display of the results, along with organization of the results in a coherent, accessible manner is vital. Optimization of parameter values to uncover (or recover) bifurcation structure is also quite important (see Chapter 3 for a discussion of this). While each tool is mostly capable of handling the numerical details of basic bifurcation analysis (continuation and bifurcation detection) in their own way, each requires that the user first find a valid steady state (or periodic solution); some (CANDYS, in particular) provide help with this. Nevertheless, the user must manually make a step to find this steady state before beginning continuation. Then, the user is required to continue a single parameter in both the increasing and decreasing directions (separately) to get just a single one-parameter bifurcation diagram. Most of the tools do have some facility to view that data graphically, but interacting with that graphic data is not possible. The real point here is that user interaction is mandatory at each step of the process. While giving the user a great deal of control over the bifurcation analysis,

this is an inappropriate model for computational biology, where we often have many dozen parameters and where small changes in one of those parameters requires us to regenerate many, many bifurcation diagrams. And this happens continuously as we adjust our models. This has been a terrible stumbling block for researchers who need to be able to gather a great deal of bifurcation information rapidly and to make decisions about how to adjust parameters to generate the desired dynamical behavior.

OSCILL8 was designed from the start to solve these problems. It is built on this model of rapid generation of bifurcation diagrams and utilizes bifurcation information in novel ways while handling the routine stuff on its own (searching for steady states, continuation in both directions, continuation all of the system parameters in succession, etc). OSCILL8 also includes novel parameter optimization algorithms that allow a user to better explore and design models. Although OSCILL8 was developed in the context of biological modeling, it is generally applicable to ODE modeling of any type. What follows is a general discussion of OSCILL8 and an exploration of various features of OSCILL8 in the form of a tutorial (though details will be left to the user documentation).

2.1 Software Architecture

OSCILL8 is written as two distinct pieces: a) a numerical core (the executable is called `o8core`) written in C++ and b) a graphical user interface (the executable is called `oscill8`) written in C# that can be run with either .NET (http://msdn.microsoft.com/netframework) or Mono (http://www.mono-project.com), making both relatively platform independent. These two components are designed as separate entities so that the numerical core can be run remotely (as a service) using powerful compute resources and the GUI can run on a local machine as a controlling client (although at the time of writing, using OSCILL8 on two different machines was not yet released for public use).

The numerical core is built on top of solid numerical code, including

- BLAS/LAPACK: linear algebra package. We use routines from this package, for example, to find steady states from which to begin bifurcation continuations. It is a very general set of mathematical routines that have been well tested for over a decade and

gives us solid ground on which to base our mathematical routines. BLAS/LAPACK can be found on the web at http://www.netlib.org/clapack.

- CVODE: efficient stiff ODE solver. This is used as the basic ODE solver engine, although OSCILL8 adds a) discrete event support (algebraic conditions involving state variables and parameters can trigger changes in state variables or parameters), b) oscillator period detection (along with minimum/maximum excursion of each state variable along an oscillatory solution) and c) steady state detection. CVODE is part of the SUNDIALS suite, found at http://www.llnl.gov/casc/sundials.

- AUTO: bifurcation continuation package. This continuation software is one of the standard bifurcation packages (Doedel et al., 1997) and can be found on the web at http://sourceforge.net/projects/auto2000. This code was modified and given an application program interface (API) so that it could be used as a library.

Each of these numerical packages has been embedded into OSCILL8 so that access to the functionality contained in each is seamless.

The graphical user interface (GUI) is built in C# using the .NET framework. Among other things, this has made development rapid and adaptable. On the down side, .NET is not yet as well supported on Linux/MacOSX environments. At the time of writing, there is a version of OSCILL8 running on Linux, but its functionality is somewhat limited. Given the (software and business community) interest in having a solid .NET platform on Linux/MacOSX, this situation is expected to improve.

2.1.1 O8W, O8R and O8M Files

OSCILL8 stores information in an XML format. The main structure relevant is the workspace, specified by the `<o8w>` tag. The user (possibly without their knowledge) creates a workspace when they analyze a model, which if done automatically by OSCILL8 will be stored in the `wscache/<model_file_name>` directory. The O8W file, as it is called, resides in the workspace directory and contains information such as the original model text used to generate the workspace (`<model_text>` element), the initial model data (i.e., values of state variables

and parameters and associate data such as whether or not it is to be considered active for a run, minimum and maximum values of the parameter etc, all stored as an `<o8m>` element) and a list of runs generated by the user (stored within a `<runs>` elements, each run with its own `<o8r>` element). The run and model data elements can be saved to separate files (the so-called O8R and O8M files) for other uses. For example, O8R files can be used directly with the command line numerical core, `o8core`, where instead of running in connection with the GUI it will run directly to the console and store the output as specified in the O8R file. O8M files can be used by `o8core` as well as the GUI to set model data before performing some specified action. Also, for the paranoid (a mighty class of folks, which when it comes to beta software, I include myself in, even if I did write it), storing model data is a way to ensure that you have parameter settings for particularly fruitful analyses that you have run. (For the paranoid, there are also a couple of useful data integrity features, see Section 2.1.2.)

These formats are constantly changing and are upgraded to meet the needs of new features and general programming considerations, so I will not detail them here (however, included in Appendix B, there are several O8R files corresponding to bifurcation matching runs, see Chapter 3). Once OSCILL8 goes into version 2, some of these details will be set in stone for compatibility reasons, and the user documentation will describe the format in more detail.

2.1.2 Data Integrity

As a group, modelers are used to bad software claiming to do things it simply cannot or *worse*, and more common, doing exactly what it should (so that it becomes quite valuable) but crashing frequently and inexplicably (and losing information gained by blood–sweat–tears). To address this concern, OSCILL8 has a couple of data integrity features:

1. Automatic backups of the workspace file (which contains all the vital information). This occurs *every* time OSCILL8 is closed, and is stored in the `backup` directory of the workspace directory, with a time-stamp appended to the workspace filename. This time-stamp is an increasing integer, so that a user can select the largest number to find the latest file. This feature is primarily useful for massive OSCILL8 crashes that end up corrupting the workspace file, rendering it unusable. The occurrence of such

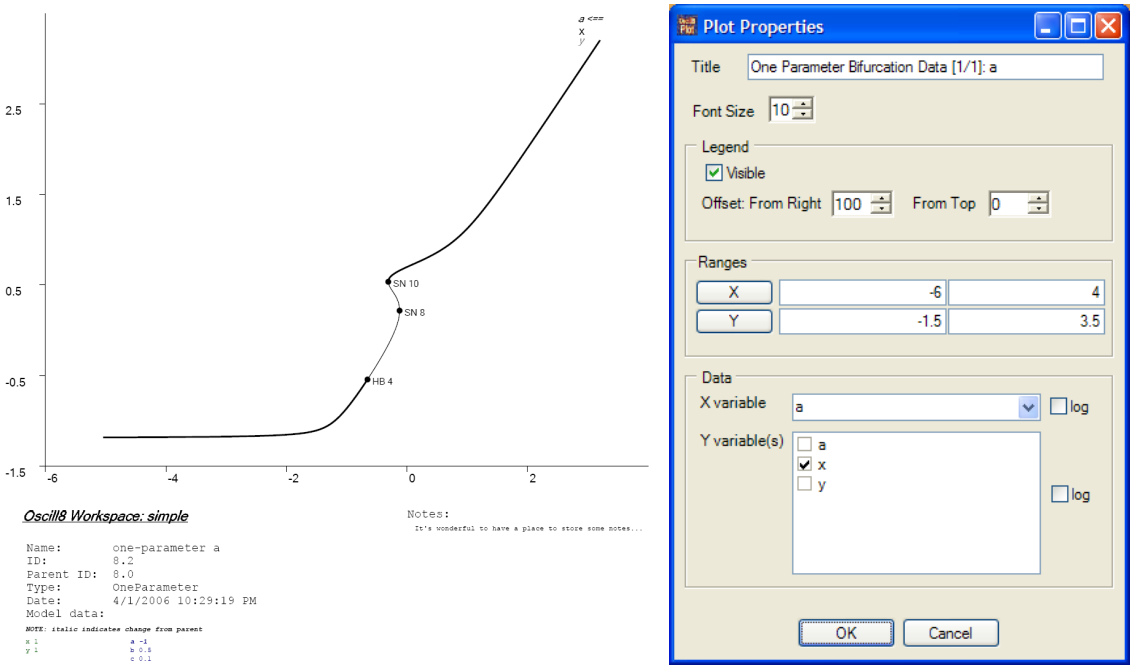
a failure, this developer hopes, is extremely rare, although when adding new features that have not been tested thoroughly by a large group of users, this kind of thing will happen from time to time.

2. Graphic regeneration. Occasionally (so far, actually, just for Kathy Chen!), saved graphic information gets corrupted somehow (this is stored as binary data, separate from the O8W file). Since the information that generated the data is okay (its stored in the O8W file), it can easily be regenerated. OSCILL8 actually detects when such data has been corrupted and will display a message in the middle of the graphic display saying "No Data/Rerun", indicating that either there is not any data or it is corrupted. Clicking Rerun (in the Run menu) will regenerate the data.

2.2 Philosophy Behind Interface

Briefly, I would like to describe the philosophy behind the user interface. For the general layout, see Figure 2.1. As mentioned, bifurcation analysis is about graphics, as is a lot of model analysis. Therefore, the choice was made early on to make OSCILL8 a graphic-centric application. Accordingly, this has been made the most prominent part of the GUI. The graphic display is made interactive by providing the ability to add and remove curves via clicking the legend, by zooming with the mouse, by alt-clicking and starting new runs from particular points within a graphic (for example, in a two-parameter bifurcation diagram, the user can alt-click at some point in the graphic and run a simulation with those parameter values). In addition, via certain keystrokes, the user can very rapidly reshape the graphic (x=zoom in x right, X=zoom out x right, Cntl-x=zoom in x left etc), save the graphic as a PNG/BMP/JPG (Cntl-S), print a summary of the current run (Cntl-Shift-P, see Figure 2.2) etc. There are many other such features and there are always new features; the user documentation provides the best information about them.

To aid a modeler in producing pretty plots (suitable for reports or publications), there are also a number of other niceties, some of which are accessible from the plot properties dialog (Figure 2.2(b)), including font size and legend placement, and directly from an alt-click on



(a) Print summary (b) Plot properties

Figure 2.2: (a) A print summary and (b) the plot properties dialog.

the legend elements themselves, where the user is given the option to change the line width and alt-width (refers to the "stable" branch of whatever is being viewed), line color and line type (currently, line type is not implemented).

The second most important aspect of the interface is the ability to organize and navigate generated results. Currently, the organization is automatic and there is a run history tree where previous diagrams and the data associated with them are organized in a hierarchical manner, where each run is a "child" of the run it was generated from (one can also display runs in a strictly chronological manner). This run history tree, which need not be displayed (for the die-hard graphic-centric user), provides a means for the user to interact with and peruse previous results. The tree also has several navigational and organizational features accessible via a context menu (alt-click in the run history area), including deletion of previous runs (along with all the children of the run) and viewing model data (values of state variables and parameters for a particular run).

A distant third in terms of importance in the interface is a place to enter notes associated with a run. This is an optional section placed just below the graphic display, and is intended to help the user remember what they were thinking when they produced a particular graphic.

This feature has turned out to be a bit less useful than originally thought, and maybe a combined notes windows where all the notes written down, organized and marked with the run name, would be more useful.

2.3 Model Analysis with OSCILL8

To begin using OSCILL8, we must first get our model into the ODE format (compatible with XPPAUT's ODE file format), which basically expresses the model equations, $\dot{x} = f(x, p)$, along with a declaration of the individual parameters that make up p and their initial values. The details of the format can be found in the online documentation. OSCILL8 is also capable of taking in models stored in SBML (Hucka et al., 2003), which is automatically converted to the ODE format by the XPPTranslator via the Systems Biology Workbench (Sauro et al., 2003). Here is what a simple (non-biological) model (adapted from Borisuk and Tyson (1998)) looks like in the ODE format (see Appendix A for a complete listing of all ODE files used in this thesis):

```
# simple, non-biological model, adapted from Borisuk & Tyson, 1998

x'=x*(1-x)*(1+x)-y
y'=(x-a)*(b-y)-c

param a=-0.5,b=0.5,c=0.1

init x=1,y=1
```

To load such a model into OSCILL8, the user can either

1. start OSCILL8, click File→Open, choose either an ODE or an SBML file,
2. start OSCILL8, click File→New, select an ODE or SBML file, adjust settings or model text in the "Create Workspace" interface,
3. or type `oscill8 model.ode` or `oscill8 model.sbml` at the command prompt.

This will bring up the main model interface (which will be empty). We will leave the gory details to the user documentation distributed with OSCILL8 (it can also be found online

at `http://oscill8.sourceforge.net`). We will, however, use this model in a moment to explore many of the following features of OSCILL8:

1. **Batched generation of one- and two-parameter bifurcation diagrams**, allowing us to get a complete picture of the local bifurcation structure for all parameters at once.

2. **Parameter "twiddler"**, which allows us to adjust parameters and see the immediate consequences to several time series plots (showing the basic waveforms of the solution) and bifurcation diagrams (see Figure 2.1).

3. **Period/min/max sampling**, which allows you to sample the period, minimum and maximum excursion of an oscillatory solution in within some interval for a particular parameter.

4. **Bifurcation search**, which allows the modeler desperate to encounter a richer region of bifurcation structure to randomly walk through parameter space and uncover bifurcations in all variables of interest.

5. **Bifurcation match/optimization**, which allows modelers to optimize parameter values to get a particular bifurcation structure related to the expected dynamics of their model (or to fine-tune parameters to adjust qualitative features related to bifurcation structure). This will be discussed in detail in Chapter 3.

6. OSCILL8 provides a number of algorithms and features related to **temperature compensation** for models where this is relevant. These will be discussed in detail in Chapter 5.

7. One of the great features of OSCILL8 is the automatic **organization** of graphic data with parameter values and algorithmic constants, which has been the bane of any modeler interested in bifurcation analysis. In addition, modelers can make notes about diagrams directly in OSCILL8's interface to keep track of thoughts related to graphics.

In addition, OSCILL8 allows the user to interact with the graphic results in an intuitive manner in many ways. For example, a context sensitive alt-click on a bifurcation point in a one-parameter bifurcation diagram allows two-parameter continuation or generation of time

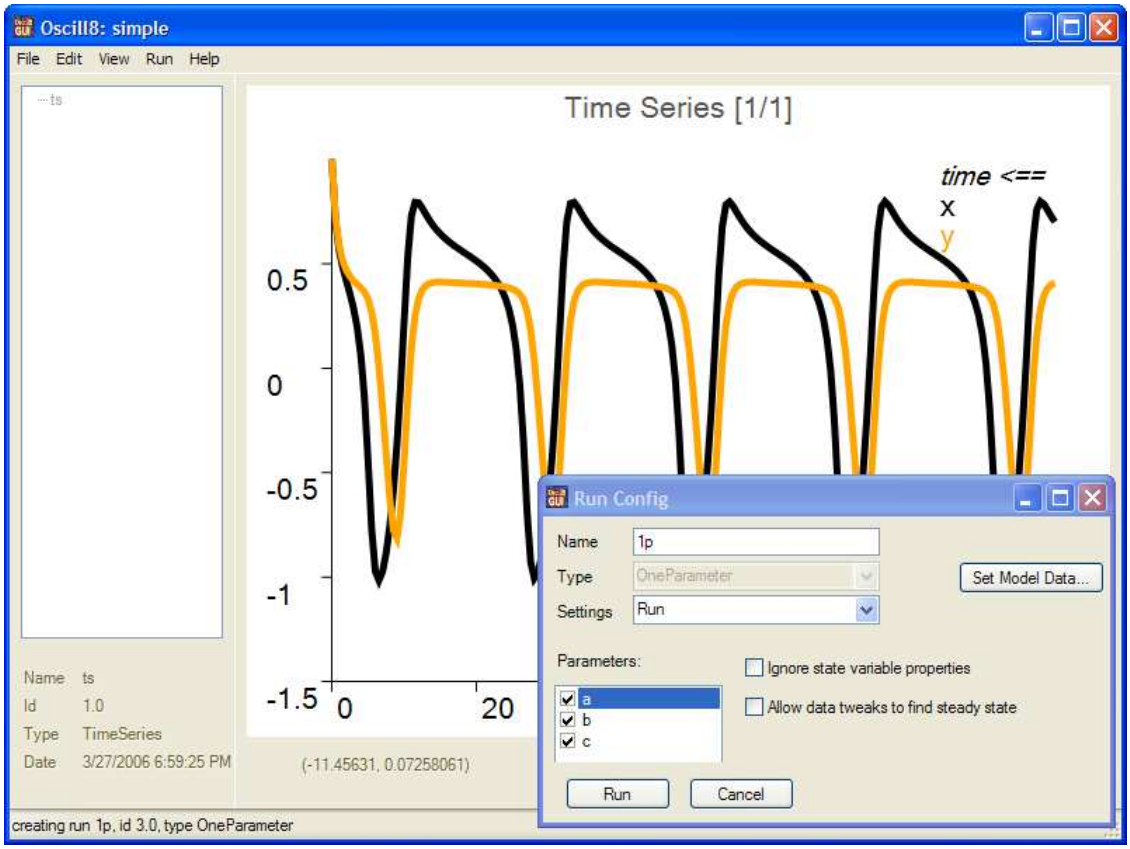


Figure 2.3: Here is how we run several one-parameter bifurcation diagrams at once.

series data for the particular value of the parameter chosen in between two Hopf bifurcations (where we might expect oscillation).

In Chapter 4, we will rely heavily on all of the features mentioned above. Items 5 and 6 will be explored in detail in Chapters 3 and 5. Let us now walk briefly through each of items 1 to 4.

2.3.1 Continuation

If you have loaded up `simple.ode` (model from above) properly and generated a single time series and then clicked on Run→1 Parameter (or equivalently alt-click in the run history area, Run→1 Parameter) you should have basically Figure 2.3 (line thickness and font size can be adjusted alt-click→Plot Properties and by alt-clicking on the individual label in the legend). In the run config window, we can see that running one-parameter bifurcation diagrams for multiple parameters is just a matter of checking a few boxes and clicking Run. When generating a one-parameter bifurcation diagram, OSCILL8 will attempt to find an

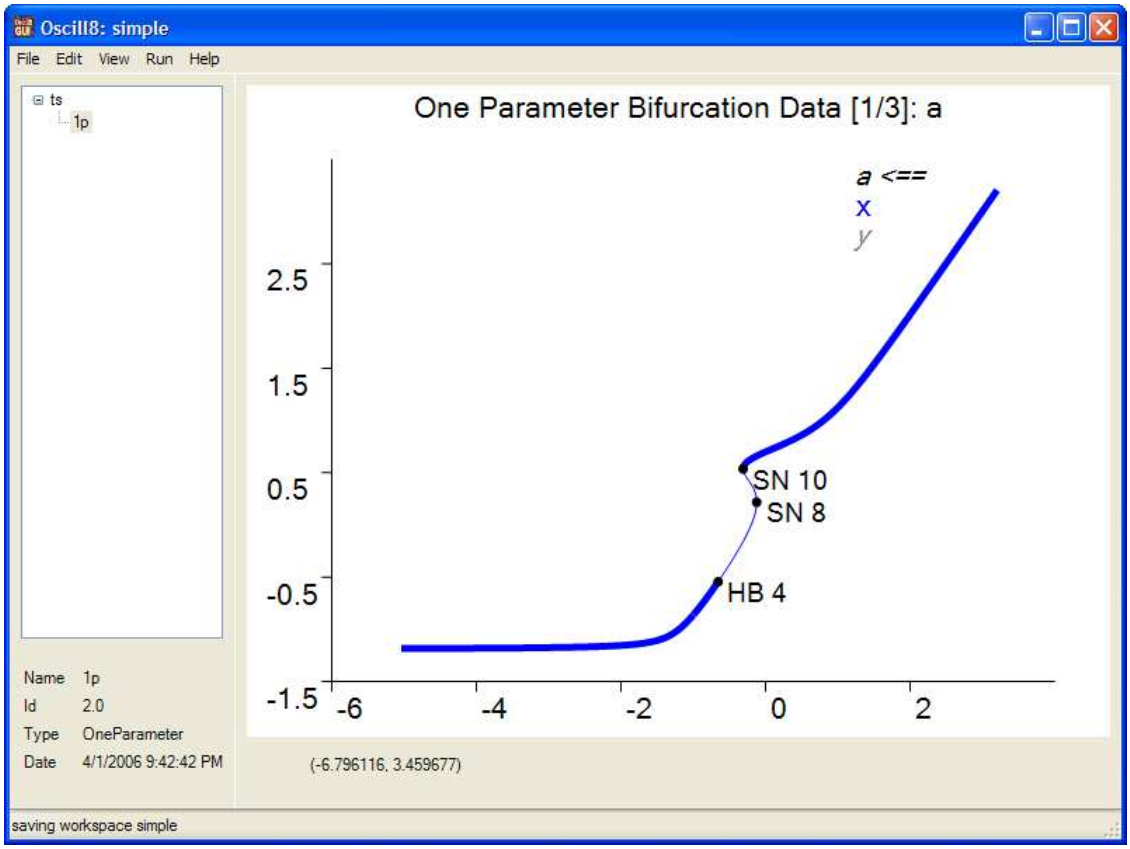


Figure 2.4: A one-parameter graph generate from the run configuration panel shown in Figure 2.3.

initial steady state by Newton's method and then by solving the differential equations to see if the system settles into a steady state. If Newton's method fails and you're in an oscillatory region of parameter space, you must change parameters and try again. This is exactly the case for our model. In order to make progress, we need to change $a = -1$ (or any similar parameter change). Such a change is made via the Set Model Data button in the run config dialog, the place where all information about parameters and state variables is kept. Then click Run. You will see a bifurcation diagram in the graphic area, labeled "One Parameter Bifurcation Data [1/3]: a", indicating that you are looking at one of three generated data sets (Figure 2.4. Clicking the up or down arrow will scroll through the bifurcation diagrams for the different parameters checked.

If you now alt-click on one of the bifurcation points in one of these diagrams, you may be given the options "Two Parameter" (for SN and HB) and "Follow Limit Cycles" (for HB). (If you have trouble getting these extra options in the menu, try zooming in on the point with the mouse first). If you select "Two Parameter" for the upper SN bifurcation

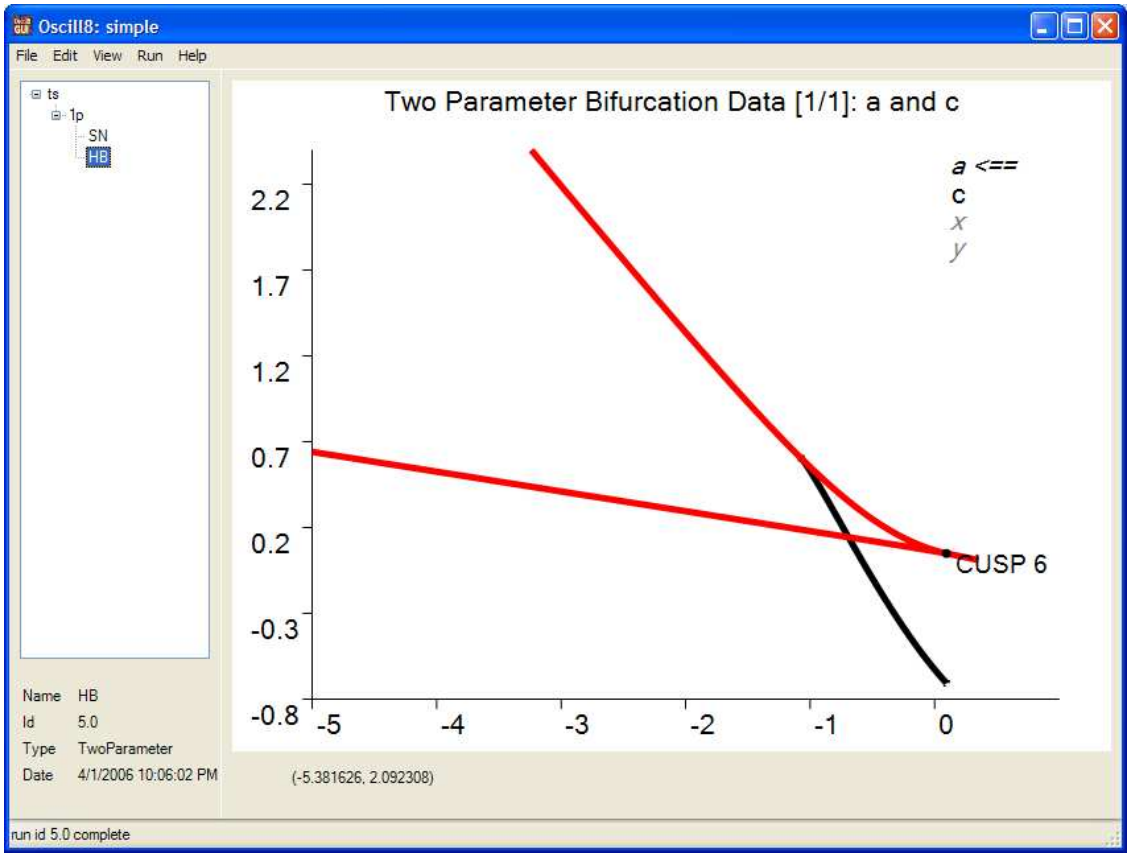
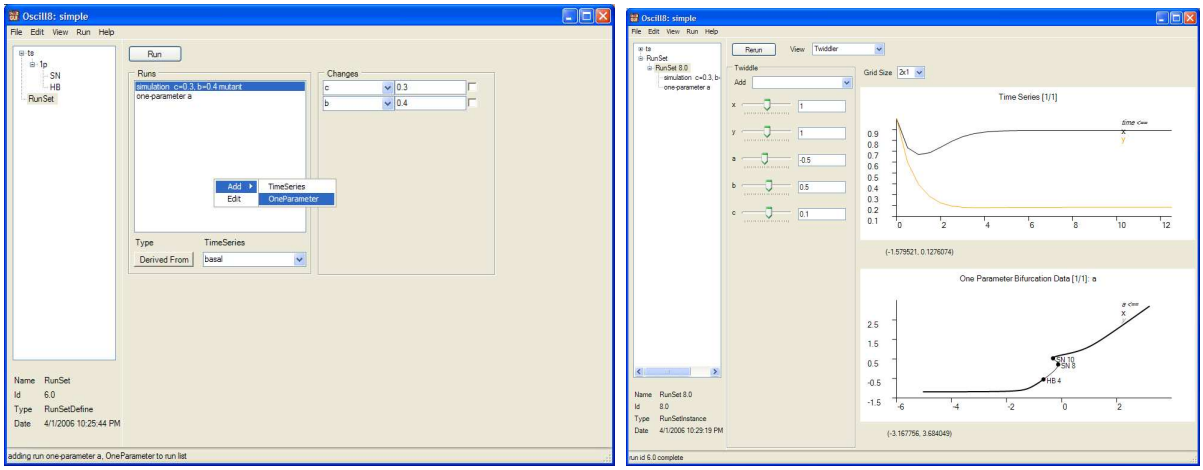


Figure 2.5: A two-parameter graph generated then combined using drag-and-drop feature. The locus of HB dies on the SN branch at a Takens-Bogdanov bifurcation point.

in the a bifurcation diagram, you will be given a another run configuration window similar to the one you got for one-parameter continuation, except this time you will see the extra options "Start Label" and "Main Parameter", which should be filled out with 10 and a (if you chose the top SN bifurcation point). The start label is visible in the graphic just after the bifurcation name (i.e., "SN 10"), so it is easy to find. If you would like to switch to another point, you can do so from the dropdown menu. Now you can select b and c to get 2 two-parameter bifurcation diagrams, b vs. a and c vs. a. If there were 20 other parameters, you could select them all and generate 20 bifurcation diagrams at once!

In addition, if you would like to combine several two-parameter plots, which makes a great deal of sense if you continue the HB bifurcation in the last example as well, you can simply drag-and-drop a previous run name from the run history area directly onto the current graph and the graph associated with the previous run will be added to the current view. See Figure 2.5 for the result of c vs. a for both the HB and SN.



(a) RunSet Definition (b) Twiddler Interface

Figure 2.6: (a) Here we see a definition of several runs that relate to important model-specific ideas, such as mutants. (b) This is the resulting twiddler interface generated from the definitions in (a).

2.3.2 Parameter Twiddler

The parameter twiddler idea is quite simple: in systems with such a high number of degrees of freedom (many parameters), it is essential to have the ability to walk our way through parameter space and evaluate the graphic data of simulation and bifurcation results. In this way, we are putting a human into the optimization loop, where the objective function is our mind. To facilitate such a mechanism, OSCILL8 provides an interface in which the user can predefine as many simulation and bifurcation diagrams as they want and then display parameter "sliders". When these parameter values are changed and Rerun is clicked, all the graphics will update to match the currently set parameters.

So the first requirement to using the parameter twiddler is predefining a "RunSet," a collection of the runs you are interested in such as "one-parameter bifurcation diagram of n, with p_1 fixed at 2.3 always" and "simulation for the mutant with $p_1 = 0$ and $n = 4$" etc. Note that each run can define any number of associated "changes," which can actually be any algebraic function of state variables and parameters (one can also use trig functions etc). Figure 2.6 shows both the RunSet definition window and the resulting twiddler interface. Next to the bifurcation matching capabilities, this has been reported as the most helpful feature in OSCILL8 to date.

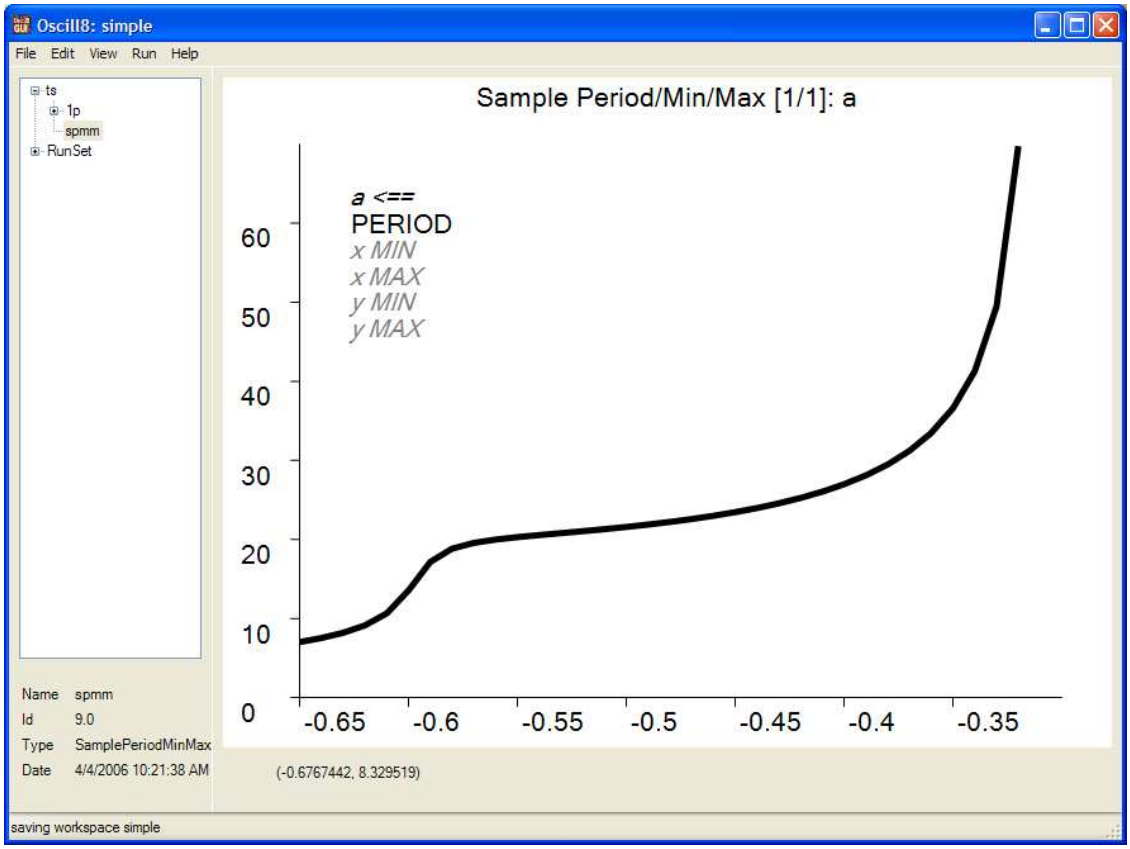


Figure 2.7: Sample period/min/max results, displaying only the period of the oscillatory solution for the basal parameter set for our simple model.

2.3.3 Period/Min/Max Sampling

It is undoubtedly useful to know the period and the minimum and maximum excursion of an oscillatory solution. When this oscillation is a limit cycle thrown off from a HB, one can use the "Follow Limit Cycle" run type which relies on the underlying AUTO engine to continue the limit cycles born near a HB. This will return a plot including the desired information. Often times, however, the oscillation may not be related to any HB for the current parameter set. In this case, we would just like to get a sampling of the period/min/max of the oscillatory solution in some range of a particular parameter. To do so, we can use the "Sample Period/Min/Max" run type. The run config dialog for this run type includes a number of parameters used to control the sampling algorithm, which is really quite simple. Basically, we compute the period/min/max and then increment the parameter by *dp* and compute again. We do this up to *n_max* times, stopping if we have failed to get a valid period/min/max sample *edge_max* times. Once we have failed going in one direction,

we repeat the process, starting with the initial parameter value and incrementing by $-dp$. The user can select which state variable to use as the basis for the period calculation via *x_index*. *period_min* and *period_max* give the min and max period to be considered, while *period_match_tol* and *period_match_max* determine how close the computed period must be from the previous cycle during the solution of the ODE to qualify as having found a valid period and how many consecutive times we need to get the same value of the period (to within *period_match_tol*) in order to have a proper sample. The remaining parameters, *h_min*, *h_max* and *h_init* control the step size of the ODE solver during the described calculations. We begin with *h_init* and once we have found what we think might be one valid period sample, we begin to use *h_max* in the region of the solution that is not close to the minimum or maximum (which are used as the anchors from which to compute period values) and *h_min* close the the anchor point. Note that these parameters can be quite important, both for speed of calculation and success of the algorithm since setting *h_max* too small will severely handicap the adaptive stiff solver algorithm (which normally would take huge jumps when it can), increasing compute time. On the other hand, setting *h_min* too large will make it impossible to get a valid sample depending on *x_match_tol* etc.

The period results for the basal parameter set using all of the default settings is given in Figure 2.7. One caveat is that the current algorithm uses *period_match_tol* as the primary driver for the solution so that sometimes the minimum and maximum for various state variables will be inaccurate. A future version of the algorithm will take this into account, given that both the period and the min/max excursion are important.

2.3.4 Bifurcation Search

This feature, which is not yet available in the GUI, is accessible via the command line. The numerical core executable, `o8core`, can be used directly by the user to perform any task the GUI is capable of via an OSCILL8 run file (O8R). (One can easily see what the O8R syntax is by looking in the OSCILL8 workspace (O8W) file for a particular model. Cutting out the text from the beginning to the end of a `<o8r>...</o8r>` block will give you an O8R file.) In addition, many common tasks have been given simpler command-line options (which the

```
> o8core -m simple.ode -bs:out bs -bs:percent 25 -bs:iterations 5

o8core, 1.20.0
author: Emery Conrad
build date: Apr  1 2006, 20:32:08

[Tue Apr 04 12:50:35 2006]: initializing the model
[Tue Apr 04 12:50:35 2006]: setting up BifurcationSearch o8r text
[Tue Apr 04 12:50:35 2006]: run o8r run text
[Tue Apr 04 12:50:35 2006]: running a bifurcation search
[Tue Apr 04 12:50:35 2006]: searching for bifurcations, 25% tweak, 5 iterations
[Tue Apr 04 12:50:35 2006]: solving for steady state...
[Tue Apr 04 12:50:35 2006]: solving time series...
[Tue Apr 04 12:50:35 2006]: searching for bifurcations...
  1: [0, a=-0.656928, 3][1, a=-0.124043, 2][2, a=-0.311526, 2][3, b=-0.17754, 2][4, ...
  2: [9, a=-5.94421, 2][10, a=-1.38242, 2][11, b=1.09298, 3][12, b=3.02797, 3][13, c...
  3: [17, a=-5.9169, 2][18, a=-1.52306, 2][19, b=1.12393, 2][20, b=1.12266, 2][21, b...
  4: [27, a=-3.2875, 2][28, a=-1.38772, 2][29, b=1.19928, 3][30, b=2.67653, 3][31, c...
  5: [35, a=-5.90209, 2][36, a=-1.52983, 2][37, b=1.10684, 2][38, b=1.0982, 2][39, b...
[Tue Apr 04 12:50:48 2006]: writing .o8m files for each bifurcation found
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 3...
[Tue Apr 04 12:50:48 2006]: run complete (13.75 s)
[Tue Apr 04 12:50:48 2006]: all done!
```

Figure 2.8: Output from a bifurcation search, a random walk through parameter space looking for bifurcations in each active parameter.

user can see by typing `o8core -help`), including the "-bs" options for bifurcation searching.

Bifurcation searching is a tool for the desperate! Imagine you have just put together a model and you know it needs to oscillate, but you cannot seem to find a HB by the normal methods (many one-parameter bifurcation diagrams etc). Then you can do a random walk in many different parameters hoping to uncover a HB! Admittedly, there may be better ways to do this (one possibility is to optimize eigenvalues to produce HB, see (Chickarmane et al., 2005)), but this tool has become surprisingly useful and likable. The random walk is controlled by a `percent` and a number of `iterations`. Each iteration, one random (multiplicative, for now) perturbation of magnitude $\in [1 - \texttt{percent}/100, 1 - \texttt{percent}/100]$ is made to each of the current parameter values.

For our model above, we know there is oscillation, but if we move ourselves outside of the region of HB for the parameter a or just consider that we might like to do some random adjustment to see what comes up, we can use the bifurcation search capability. Running `o8core` on this model gives the output seen in Figure 2.8. Let us decode it! After initializing etc, we begin to see indented lines starting with `#:`, where # is the iteration

number. Following that, we see the results of a one-parameter bifurcation diagram for every variable that is active (which, in this case, is all of them since I did not specify an O8M file for input—for details of the O8M file, see Section 2.1.1). During iteration 1, which uses the unperturbed basal set, we see `[0, a=-0.656928, 3][1, a=-0.124043, 2][2, a=-0.311526, 2]` first. Each bracketed set is a bifurcation point: the first number, `0`, is the overall index to distinguish this bifurcation point from others in the output, the second element, `a=-0.656928`, is the value at which the bifurcation occurs and the third element, `3`, is the type of the bifurcation (2 = SN, 3 = HB, following AUTO's conventions). This completes the bifurcation diagram for a in the first iteration. Following this is similar information for each of the active parameters (lines are truncated here for convenience).

We can see the effects of the single vector perturbation each iteration by the changing location of the bifurcation points: note that on iteration 2, we lose the initial HB (type = 3) and only have the two SN bifurcations, which have moved considerably. In this way, we can perturb parameters about and get a feel for local bifurcation space. If we like a particular bifurcation point and we would like to start model analysis from that point, we need only look for the output file `<bs:out>_<index>.o8m`. That is, we look a file with the name starting with whatever we specified via the command line option `-bs:out` followed by an underscore followed by the index number, as just described. For example, the first bifurcation point for iteration 3 has index 17, and we specified `-bs:out bs` on the command-line, so if we would like to start a model analysis from that point, we would need the `bs_17.o8m` file, which looks like:

```
<o8mi>
  <xdim>2</xdim>
  <species name="x" active="True" value="0.570501" min="-1000" max="1000" type="1" />
  <species name="y" active="True" value="0.384819" min="-1000" max="1000" type="1" />
  <pdim>3</pdim>
  <par name="a" active="True" value="-5.9169" min="-10" max="10" eps="0" s="0" type="6" />
  <par name="b" active="True" value="0.537824" min="-5" max="5" eps="0" s="0" type="6" />
  <par name="c" active="True" value="0.992605" min="-1" max="1" eps="0" s="0" type="6" />
  <vdim>0</vdim>
  <adim>0</adim>
</o8mi>
```

This file can be loaded into an OSCILL8 workspace via the Model Data, Read button to initialize the model to be at the bifurcation point. At some point in the future, the bifurca-

tion searching functionality will be integrated directly into the GUI, along with bifurcation matching etc, so these details may be less relevant.

2.4 Current Status, Limitations and Future Directions

OSCILL8 is currently available online at `http://sf.net/projects/oscill8` and is open source. It has reached a fairly stable version and is being used by several labs interested in modeling biological processes. Development will continue for the foreseeable future, including better integration with SBW and addition of the search and optimization features (of Chapter 3) to the GUI.

Currently, the bifurcation searching and matching (i.e., parameter optimization) features are only available via the command line, and are essentially experimental. This will be addressed in future versions. There is a (growing) list of bugs and inconveniences at sourceforge which will be maintained and addressed, as long as development continues. The prospects are good for the tool to be around for some time (in some form), since the author has begun to rely quite heavily on it to do any and all of his dynamical systems research.

Chapter 3

Bifurcation Matching

Formulating valid mathematical models is a tricky business, especially with respect to identifying a valid set of system parameters. Typically, methods of parameter estimation rely on time series data or information derived from time series data (i.e., solutions of the ordinary differential equations). However, another primary concern in properly fitting a model is to realize or retain a particular qualitative dynamical behavior, often represented by a bifurcation diagram. This is the case, for example, for the cell cycle, which relies on oscillation derived from a hysteresis loop rather than a limit cycle. This hysteresis can be directly linked with physiological phenomena, and therefore it is necessary to realize the dynamics represented by the bifurcation diagram when developing a new model or to conserve this diagram when adding complexity to an existing model. Another point to note is that, because we generally lack sufficient experimental data, fitting a model to time series data may not be sufficient to realize the bifurcation structure. It is important, therefore, to develop a general means for incorporating bifurcation structure into a model fitting scheme.

In this chapter we will be concerned with reproducing the dynamical properties of the steady state curve in a *one-parameter bifurcation diagram*, so we will generally drop the "one-parameter" label when referring to bifurcation diagrams and bifurcation structure. Note that for now we do not explicitly consider any curves other than the steady state curve—limit cycle continuation is not included as an explicit target—during optimization, although the existence of limit cycle oscillation is often implicit in the relative layout of the bifurcation points in a diagram. An obvious extension to this work might include limit cycle

continuation, two-parameter continuation curves and associated codimension 2 bifurcation points, and even optimization of several one- or two-parameter bifurcation diagrams at the same time.

3.1 Overview

Let us start with a few definitions relevant to the optimization of bifurcation information.

Definition 3.1. *A* **model** *is a system of nonlinear ODEs given by*

$$\dot{x} = f(x, p), \qquad x \in \mathbb{R}^n, p \in \mathbb{R}^m \tag{3.1}$$

with the same properties as for system (1.1).

The important point here is that the idea of a model includes the freedom represented by p, rather than specifying a particular fixed value for p.

As described earlier (see Section 1.2), a local bifurcation occurs as we move through parameter space if Re λ passes through zero (referred to earlier as condition (1.2)) for some eigenvalue, λ, of the Jacobian matrix, $f_x(x, p)\big|_{(x^*,p^*)}$, where (x^*, p^*) is some steady state. For the purposes of optimization, we identify a bifurcation point with a triple

Definition 3.2. $B = (\tau, p^*, x^*) \in (\mathbb{T}, \mathbb{R}^m, \mathbb{R}^n)$ *is called a* **bifurcation point** *assuming condition* (1.2) *is met, where* $\mathbb{T} = \{SN, HB, \dots\}$ *is the set of possible bifurcation types under consideration and* (x^*,p^*) *is some steady state of of system* (3.1).

Depending on context, we may simplify the notation by treating p^* as a scalar (since we are only going to be considering one-parameter bifurcation diagrams where we concentrate on a single parameter) and we may only consider one dimension of x^* to match with a particular 2-D plot.

Next, let us be clear and make a distinction between a one-parameter bifurcation diagram and *bifurcation structure*. A bifurcation diagram is a picture rich with content; it depicts how the steady state values and other dynamical quantities of a particular state variable depend on a particular parameter. This diagram includes not only the sequence of bifurcation

values along the steady state curve, but also the geometry of the curve itself, not to mention other related dynamical information (e.g., maximum and minimum values of a limit cycle oscillation, etc). To keep things simple and to be precise, we refer to the essential sequence of bifurcation points not as a bifurcation diagram, but as

Definition 3.3. *A* **one-parameter bifurcation structure** *for a particular parameter, p_i, is an ordered collection of bifurcation points, $\{B_j\}$, where the order is defined by tracing out the steady state curve from some minimum value of p_i and recording bifurcations encountered along the curve. We will formally write the bifurcation structure as*

$$S_{p_i} = \{B_j\}_{j=1}^{N} = \{(\tau_1, \alpha_1, \beta_1), (\tau_2, \alpha_2, \beta_2), \ldots, (\tau_N, \alpha_N, \beta_N)\} \tag{3.2}$$

where α_j is a particular value of p_i, β_j is a particular value of x (possibly a vector quantity), and τ_j is a the corresponding type of the bifurcation point found at (α_j, β_j). Note that α_j is necessarily scalar, whereas, depending on the context, β_j may be either a scalar or vector quantity. Two bifurcation structures are called **compatible** *if they have the same number of bifurcation points and the sequence of types, τ_i, are identical.*

Although an abuse of notation, we may also refer to the bifurcation structure as simply the sequence of τ_j, as this is often the only important point of interest in context. This, of course, makes a *big* difference when we deal with the question: how do we optimize parameters to reproduce a particular bifurcation structure? It also reveals that we are juggling the desires to optimize both qualitative and quantitative properties of dynamical structure. Also note that because we are primarily interested in non-branching bifurcations in this thesis, we can either ignore the fact that the "order" of the sequence of bifurcation points may not be well-defined (e.g., at a pitchfork bifurcation, there is more than one curve that can be followed) or assume that the continuation algorithm will choose to follow the branches in some well-defined manner (e.g., if we are interested in a bifurcation diagram of x_j vs. p_k, then we might require that branches are followed in the order of increasing values of x_j).

Generally speaking, given a model (M), a parameter of special interest (p_i) and an initial and target bifurcation structure (S and $\hat{S}$), we would like to adjust the system parameters to reproduce $\hat{S}$ starting from S. Since the end goal is often to reproduce dynamical behavior

associated with $\hat{S}$, sometimes what we would really like to do is *qualitatively* match a bifurcation diagram or even a class of bifurcation diagrams that give similar dynamics. On the other hand, especially when we have more data and have already fit our model to that data, we would like to match quantitative details of the diagram so as not to disturb carefully selected quantitative features of the solution. The algorithm we develop should keep this in mind. Another unpleasant reality is that the problem may not have a valid solution. Although the systems we are typically interested in include both positive and negative feedback loops and the interactions of the network are likely to produce SN, HB and SNIC bifurcations, we cannot guarantee whether or not a particular bifurcation structure is possible to reproduce. This is not a question we will answer in this work, but this issue would be an extremely useful (and fruitful, I expect) line of research to pursue.

3.1.1 A Simple Model

The algorithm by which we can solve the optimization problem will be presented below in Section 3.2, but let us first explore the problem a bit further in the context of a simple (non-biological) model that has all the bifurcation structure of interest to us (taken from Borisuk and Tyson (1998)):

$$\begin{aligned} \dot{x} &= p\,[x(1-x)(1+x) - y] \\ \dot{y} &= (x-a)(b-y) - c \end{aligned} \tag{3.3}$$

where p, a, b and c are our system parameters with base values of $p = 1$, $a = -0.5$, $b = 0.5$ and $c = 0.1$. Solving $\dot{x} = 0$ and $\dot{y} = 0$ produces the steady state equation (assuming $p \neq 0$):

$$x^4 - ax^3 - x^2 + (a+b)x - ab - c = 0 \tag{3.4}$$

Note that p has no bearing on the steady state curves (assuming $p \neq 0$) but does have a bearing on the stability, and therefore bifurcation values, along the curve. Also, let us define three bifurcation structures for this model to use as examples in the development of the optimization algorithm:

$$S_0 = \{(HB, -0.66, -0.53), (SN, -0.13, 0.23), (SN, -0.31, 0.54)\}$$

$$S_1 = \{(HB, -1.07, -0.25), (SN, -1.0, 0), (SN, -1.1, 0.45)\}$$

$$S_2 = \{(SN, 0.61, 0.95), (SN, -2.32, -0.6), (SN, -1.37, -1.1)\}$$

(see Figures 3.1 and 3.3 for these structures and the associated parameter values) where we are using the abbreviated forms matching just the details of the 2-D data. Along with these three structures, let us define three optimization problems

$$\text{T1: } S_0 \longrightarrow S_1$$

$$\text{T2: } S_0 \longrightarrow S_2$$

$$\text{T3: } S_2 \longrightarrow S_0$$

In addition, a number of variant of optimizations T1, T2 and T3 were run to exercise various parts of the algorithm. These are T1-P, T1-R, T2-R and T3-R, where the "-P" stands for an optimization where p was held fixed and "-R" stands for an optimization based on relative fitness (see Section 3.1.3).

3.1.2 Compatible Structures: Problem T1

Now, consider Figure 3.1, where we have depicted initial and target bifurcation structures for problem T1, S_0 and S_1. We can intuitively say that S_0 and S_1 are "close" to each other, but we need a quantitative measure of this intuition. If two structures have the same sequence of bifurcation types, as do S_0 and S_1 (i.e., $\{HB, SN, SN\}$) we can simply measure the distance from one to the other (or the fitness of one with respect to the other) in the normal least-squares way:

Definition 3.4. *The* **simple fitness** *or* **simple objective function** Φ^* *of a bifurcation structure* $S = \{(\tau_j, \alpha_j, \beta_j)\}_{j=1}^N$ *with respect to* $\hat{S} = \{(\hat{\tau}_j, \hat{\alpha}_j, \hat{\beta}_j)\}_{j=1}^N$ *is, assuming that* $\tau_j =$

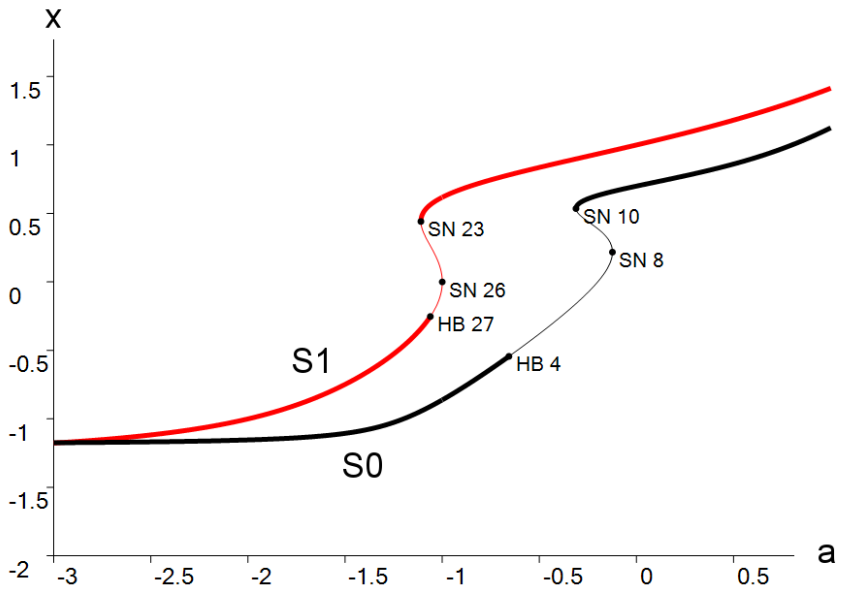


Figure 3.1: Bifurcation structures S_0 and S_1 for model (3.3). The relevant parameter values for S_0 are $p = 1$, $b = 0.5$ and $c = 0.1$, while for S_1 we have $p = b = c = 1$. Optimization problem T1 adjusts the parameters from S_0 values to S_1 values. These two bifurcation structures are "close" in the sense that the sequence of bifurcation points have the same types: $\{HB, SN, SN\}$, making problem T1 a fairly straight forward optimization problem.

$\hat{\tau}_j, j = 1, \ldots N$, *defined as*

$$\begin{aligned}\Phi^*(S, \hat{S}) &= \sum_{j=1}^{N} \Phi^*(B_j, \hat{B}_j) \\ &= \sum_{j=1}^{N} \omega_j \cdot \left[(\alpha_j - \hat{\alpha}_j)^2 + \sum_{k=1}^{n} (\beta_{j,k} - \hat{\beta}_{j,k})^2 \right]. \end{aligned} \tag{3.5}$$

where ω_j is a point-specific weight. We say that S and $\hat{S}$ are Φ^***-close** *if $\Phi^*(S, \hat{S})$ is small.*

Note that we may not include $\beta_{j,k}$ for all of the state variables (often, we use just one to match a particular 2-D plot) and that two structures with differing sequences of bifurcation types cannot be adequately compared with Φ^*. Φ^* works well if our target structure is quantitative, in the sense that we are after the exact bifurcation structure $\hat{S}$ and not the qualitative dynamical behavior represented by $\hat{S}$. We will address the latter situation in the next section.

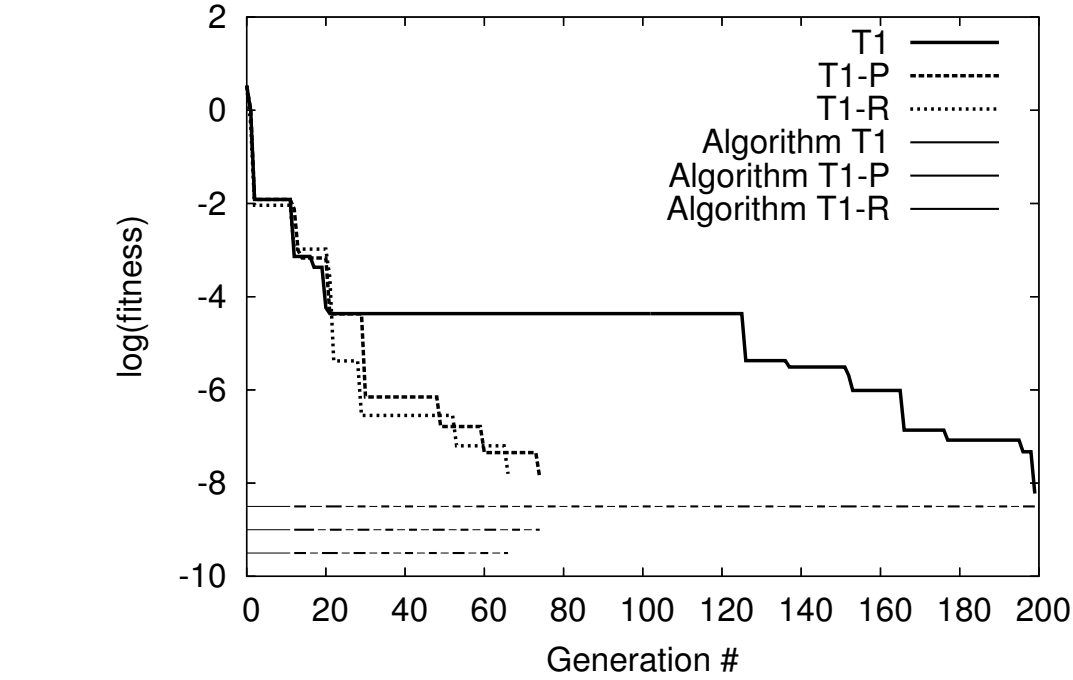
The fact that S_0 and S_1 are Φ^*-close makes the optimization for T1 fairly routine, because the objective function is not only continuous, but smooth, within the boundaries of the

local manifold defined by $\{HB, SN, SN\}$. Theoretically speaking, T1 can be solved by a standard optimization algorithm relying on local derivative calculations of Φ^* with respect to each parameter in order to make steady progress toward an optimum. The algorithm described below for solving bifurcation structure optimization has a mode that uses derivative information, though the main aim of this matching algorithm will be to overcome other issues as described below in Section 3.1.3. The fitness values for the optimization of T1 can be seen in Figure 3.2(a) and a fitness landscape (3D plot of fitness vs. $b \times c$) in Figure 3.9.

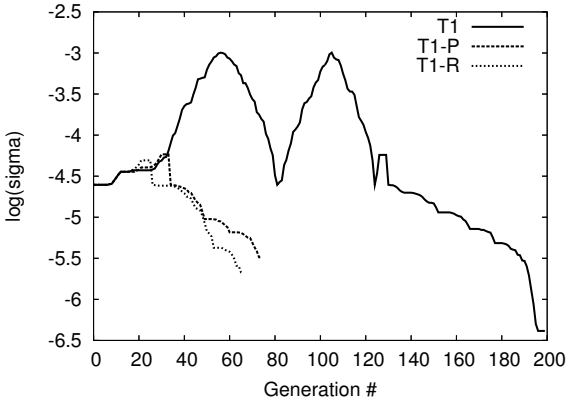
3.1.3 Incompatible Structures: Problems T2 and T3

Problem T2, on the other hand, is of a somewhat more complex nature. Figure 3.3 shows S_0 and S_2, which are clearly not close, at least in any obvious sense. Indeed, S_2 has a different sequence of bifurcation types than S_0 $\{SN, SN, SN\}$, and it is not clear which pair of the SN bifurcations from S_2 correspond to the pair in S_0, if any. We would also have a hard time deciding what metric is appropriate to measure the distance between these two structures because clearly Φ^* is not appropriate (it is not even defined). We could try to define a more complex metric to compare more general bifurcation structures, for example, based on eigenvalues and their proximity to certain types of bifurcations, but the simplicity of utilizing a metric like Φ^* is enticing.

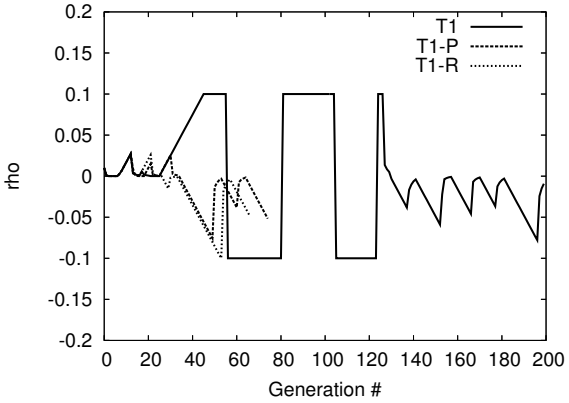
The alternative to making the metric more complex is to design an algorithm that addresses this complexity. Indeed, we *can* imagine a sequence of intermediate bifurcation structures, each "close" to one another in some sense (e.g., one extra bifurcation type or point away from each other), forming a bridge between S_0 and S_2 that could be used as stepping stones during an optimization. Even if our imagination is not working so well, we can do preliminary bifurcation analysis close by the initial structure (using the *parameter twiddler* feature of OSCILL8, see Chapter 2) to explore possibilities (which of course would mean that we have already done some of the optimization by hand!). In this case, such an analysis would rapidly uncover several intermediate structures suitable for our purposes, including $\{HB, HB\}$ (where the top two SN bifurcations disappear in favor of a HB) and $\{HB, HB, SN, SN\}$. Also, we recall that pairs of SN bifurcation often collide and disappear



(a) Fitness during run



(b) σ variation



(c) ρ variation

Figure 3.2: T1 Results. a) The upper 3 curves depict the progress of the fitness during optimizations T1, T1-P and T1-R while the curves at the bottom of the graph indicate when we were using the single stepping algorithm (thin line) or the vector perturbation algorithm (thick line). b) σ is adapted according to ρ in c) which is modified based on the progress seen in a). Note also that these optimization runs fairly evenly split their time between the single-step and vector perturbation algorithms. Compare this to T3, which spent the majority of time in single-step mode.

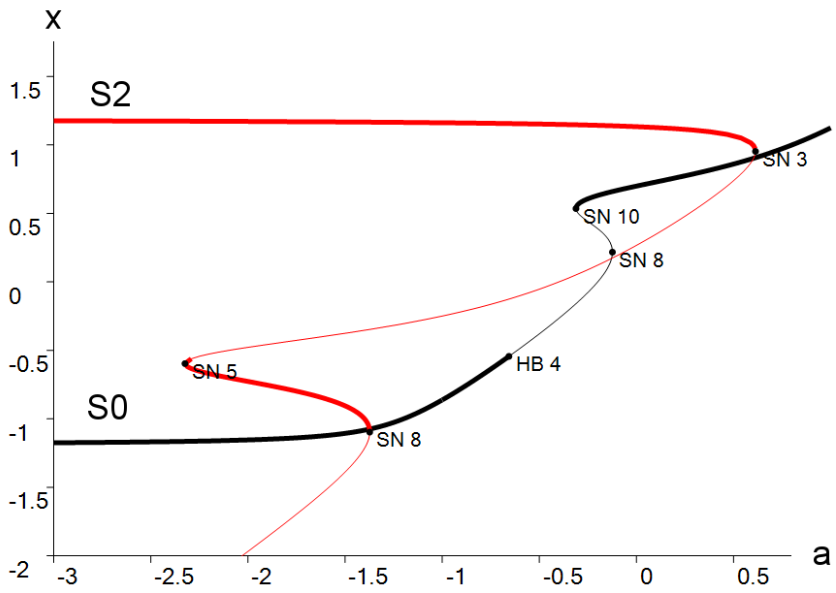
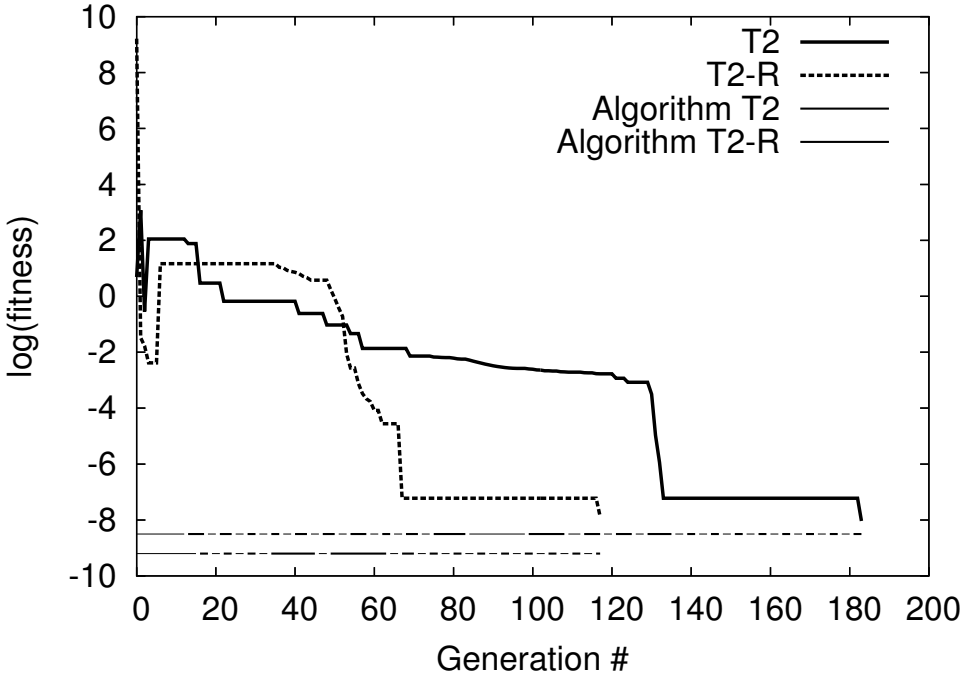


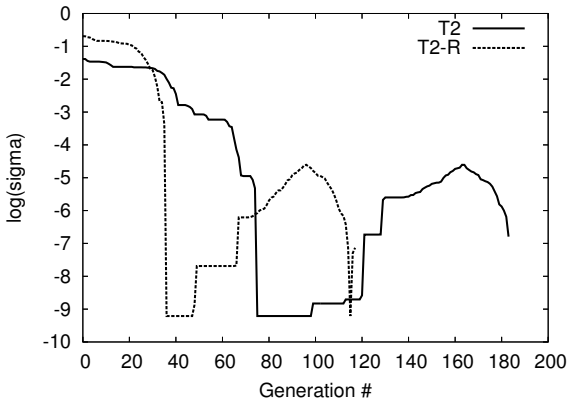
Figure 3.3: Bifurcation structures S_0 and S_2 for model (3.3). The relevant parameter values for S_0 are $p = 1$, $b = 0.5$ and $c = 0.1$, while for S_2 we have $p = 1$, $b = -0.5$, and $c = -0.2$. Optimization problem T2 adjusts the model from S_2 to S_0. These two bifurcation structures are not "close" to each other because the sequence of bifurcation points have different types: $S_0 = \{HB, SN, SN\}$ while $S_2 = \{SN, SN, SN\}$.

at a cusp bifurcation, indicating that $\{SN\}$ and $\{SN, SN\}$ might be good intermediates as well. Given these intermediate structures, we can begin an optimization at S_0 and generate new test sample points in parameter space, using Φ^* to compare compatible bifurcation structures and trying to hop from one intermediate to the next in an attempt to recover a structure compatible with S_2. The next obvious question is: how do we jump from one intermediate to the other?

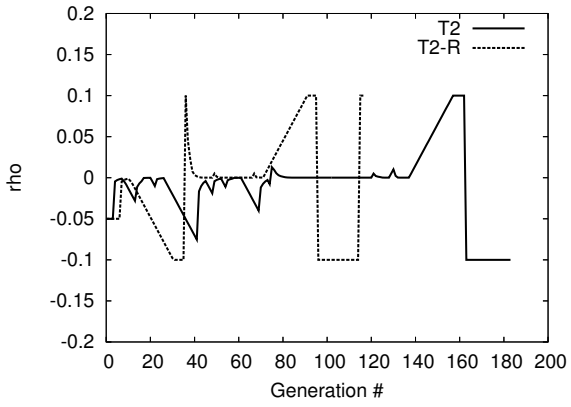
There is an easy way, which we have implemented in the current algorithm, and a more rigorous way, which will be a topic of future work. The easy way relies on random perturbations in parameter space which attempt to discover the boundaries between incompatible bifurcation structures. The good news about this is that it is computationally efficient (requires NO computation, just selection of random numbers) and if you have chosen a smart set of intermediate structures that really are close to each other, jumping from one to the other via random perturbation should be relatively successful. The downside of this method is that there is no guarantee that you will be "lucky" enough to perturb the structure across a boundary. For the models this algorithm has been tested on to date, this has not been an issue. In the future, this will need to be investigated.



(a) Fitness during run

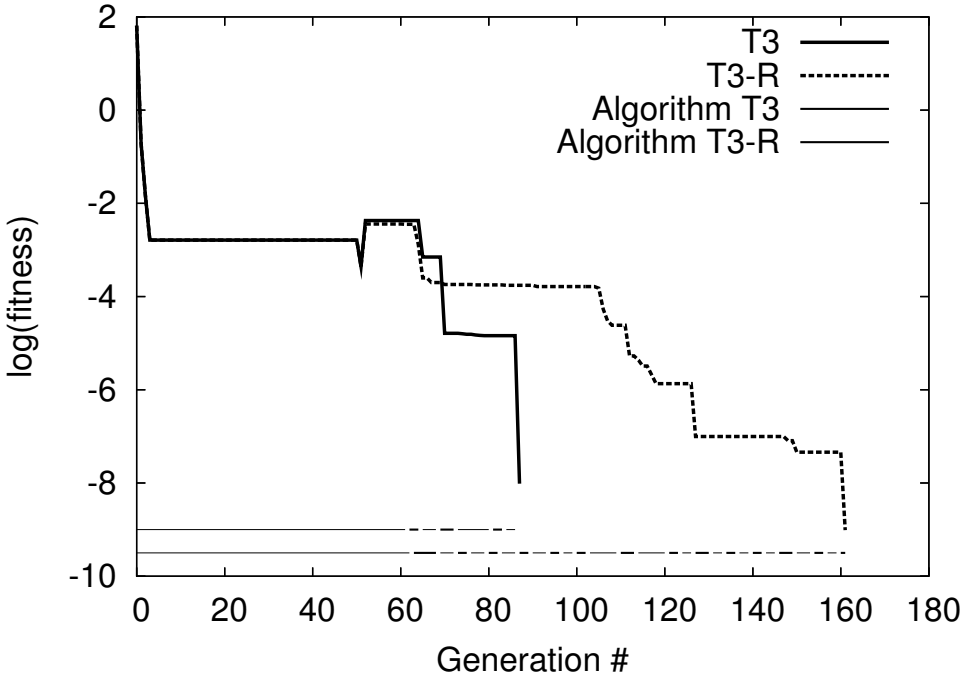


(b) σ variation

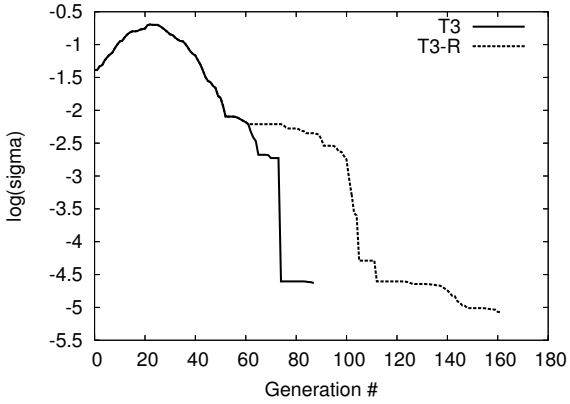


(c) ρ variation

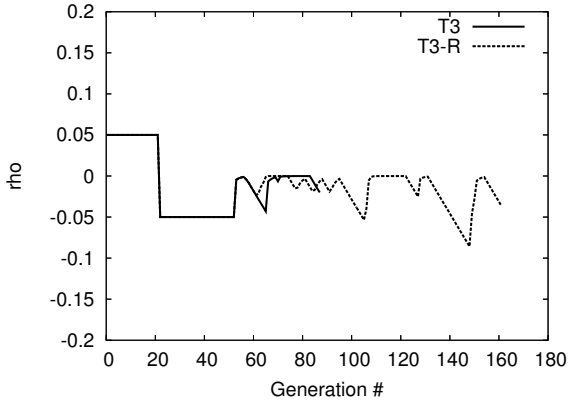
Figure 3.4: T2 Results. See description in Figure 3.2 and discussion in the text. Note here that T2 spends much more time than the T1 or T3 optimizations in the vector perturbation algorithm, evident in the 3D sample Figure 3.11, which has a "cloud" of points instead of the structure seen in Figure 3.12 for T3.



(a) Fitness during run



(b) σ variation



(c) ρ variation

Figure 3.5: T3 Results. See description in Figure 3.2 and discussion in the text. On point to note here is that other than the startup for both T3 and T3-R, fairly steady progress was made since $\rho_{max} = 0.1$ was never reached (if progress stalls for very long, ρ will reach $\pm\rho_{max}$. Notice also that these optimizations spent most of their time in single-step mode, which differs from the T1 and T2 sets. This may be due to the restriction of parameter ranges which helped the algorithm to hone in better. The fact that we spend most of our time in single-stepping mode is evident in the 3D plots of sample points for T2 and T3 seen in Figures 3.11 and 3.12. Note the linear structure of the T3 sample set, indicating progress in one direction for many iterations of the single-stepping algorithm.

3.1.4 Relative Fitness

One last issue to consider is that we are often more interested in certain qualitative features of a bifurcation diagram. This is especially true when we are just exploring candidate models early in the modeling process. A good example of this is S_0 above. Suppose we are modeling some process that requires a stable steady state for large a but oscillation for some intermediate region of a values. Furthermore, we would like the transition from the high-a stable steady state to the oscillatory domain to be abrupt and have large amplitude oscillation associated with it (via a SNIC bifurcation, which is what we have here). This qualitative description could be represented by a number of incompatible structures, including S_0 (or, for example, the transition from steady state to large amplitude oscillation could occur via a subcritical HB). In addition, the quantities at which the bifurcations occur, i.e., (α_i, β_i), are relatively less important than the qualitative behavior, represented by the relative position of these points. This relativity amounts to scaling the axes of the diagram. Therefore, we are motivated to modify our simple fitness function Φ^* to handle this concept of relativity:

Definition 3.5. *The* **relative fitness** *or* **relative objective function** Φ *of a bifurcation structure* $S = \{(\tau_j, \alpha_j, \beta_j)\}_{j=1}^N$ *with respect to* $\hat{S} = \{(\hat{\tau}_j, \hat{\alpha}_j, \hat{\beta}_j)\}_{j=1}^N$ *is defined as (still assuming* $\tau_j = \hat{\tau}_j, j = 1, \ldots N$*)*

$$\begin{aligned}\Phi(S, \hat{S}) &= \sum_{j=1}^{N} \Phi(B_j, \hat{B}_j) \\ &= \sum_{j=1}^{N} \omega_j \cdot \left[\left(\frac{\alpha_j}{|\alpha|} - \hat{\alpha}_j \right)^2 + \sum_{k=1}^{n} \left(\frac{\beta_{j,k}}{|\beta_k|} - \hat{\beta}_{j,k} \right)^2 \right]. \end{aligned} \tag{3.6}$$

where $|\alpha| = \alpha_1/\hat{\alpha}_1$ *and* $|\beta_k| = \beta_{1,k}/\hat{\beta}_{1,k}$*. By* $\Phi_{\alpha,k}$*, we mean* Φ *with* $|\beta_k| \equiv 1$ *and by* Φ_β *we mean* Φ *with* $|\alpha| \equiv 1$*. Note that* $|\alpha|$ *and* $|\beta_k|$ *are chosen such that the sum over* $j = 1, \ldots, N$ *could be taken from* $j = 2, \ldots, N$ *(this would not be the case for* $\Phi_{\alpha,k}$ *or* Φ_β*). If we set both* $|\alpha| \equiv 1$ *and* $|\beta_k| \equiv 1\ \forall k$*, then we get* Φ^**. Again, we say that* S *and* $\hat{S}$ *are* **Φ-close** *if* $\Phi(S, \hat{S})$ *is small.*

Essentially, Φ scales the axes so that the first bifurcation point of a sample structure that is being tested against a target is correct; other bifurcation points are adjusted so that it is

only their relative values that matter.

3.2 Algorithm

Now that we appreciate the problem a bit better, let us design an algorithm to solve it. First, because generating bifurcation diagrams for a particular set of parameters can be computationally costly (at least relatively speaking), we would like to devise an algorithm that does not require excessive sample point evaluation. Second, we are typically dealing with very high dimensional parameter space, so testing the derivative with respect to each parameter may be too costly. These two factors by themselves motivate the use of random perturbations, and cog nicely with the issue discussed earlier about trying to "jump" from one intermediate structure to another (when initial and target structures are incompatible), so we will focus on random perturbation for now.

By **sample** or **sample point**, we mean a particular value in parameter space $p^* \in \mathbb{R}^m$. Each sample point has an associated bifurcation structure that must be computed. For each intermediate or target structure, S_i, we will evolve a separate population of sample points, P_i, as if it were its own optimization problem targeting the corresponding structure, with the caveat that sample points that are generated and deemed to belong to a different P_i will be moved to that population. Each iteration of the algorithm consists of generating a number of new sample points that are perturbations of samples from each P_i, which are subsequently classified (i.e., to which P_i does it belong?) and fitness–evaluated with respect to the corresponding structure. In this way we hope to make progress toward a defined goal.

There are two separate algorithms for generating perturbed samples:

1. A **full random vector perturbation** of a sample point p^* is generated by selecting m random numbers, r_i. from some distribution and perturbing each p_i^* value in either a multiplicative or additive manner (see the following discussion).

2. A **single random parameter perturbation** of the best sample point, p^*, is generated by selecting a particular parameter, p_i^*, and perturbing it up or down either multiplicatively or additively (again, discussion follows).

Require:
p^*=sample to perturb
$\mathcal{D}$=either $\mathcal{U}$ or $\mathcal{N}$ distribution, according to T_p
j is current single index if $T_p = 2$
$k = -1$ or 1, toggled each call when $A = 2$

1: **if** $A = 1$ **then** // *vector*
2: **if** $T_p = 1$ **then** // *multiplicative*
3: **choose** $r_i \in \mathcal{D}(1, \sigma), i = 1, \ldots, m$
4: **set each** $p_i^* \leftarrow r_i \cdot p_i^*$
5: **else if** $T_p = 2$ **then** // *additive*
6: **choose** $r_i \in \mathcal{D}(0, d_\sigma \cdot \sigma), i = 1, \ldots, m$
7: **set each** $p_i^* \leftarrow p_i^* + r_i$
8: **end if**
9: **else if** $A = 2$ **then** // *single*
10: **if** $T_p = 1$ **then** // *multiplicative*
11: **set** $p_j^* \leftarrow (1 + k \cdot \sigma) \cdot p_j^*$
12: **else if** $T_p = 2$ **then** // *additive*
13: **set** $p_j^* \leftarrow p_j^* + k \cdot d_\sigma \cdot \sigma$
14: **end if**
15: **end if**
16: **return** p^*

Figure 3.6: Sample generation algorithm, depending on the algorithm, A, and the perturbation type, T_p (see Table 3.3). $\mathcal{D}$ is either a uniform ($\mathcal{U}$) or normal ($\mathcal{N}$) distribution, where $\mathcal{U}(c, d)$ indicates a uniform distribution of length $2d$ centered at c and $\mathcal{N}(\mu, \sigma)$ indicates the normal distribution with mean μ and standard deviation σ.

In the first case, we are attempting to make random jumps around parameter space in an evolutionary scheme, whereas in the second case, we are basically sampling local derivatives for each parameter in order to make a small step toward the goal. See Figure 3.6 for the exact algorithm used. In the implementation of this algorithm in OSCILL8, the algorithm is a user selectable feature via the `<algorithm>` parameter (see Table 3.3).

3.2.1 Multiplicative vs. Additive Perturbation

As mentioned, perturbation can be made in a multiplicative or additive manner. In biological modeling, we are often dealing with the constraint that all parameters and state variables are positive quantities, making **multiplicative perturbation** a natural choice. This method also has the advantage of disregarding magnitudes with regard to the amount of space searched. That is, if 1 is as likely a value as 100, the multiplicative perturbation will search

more sparsely about 100 for a given perturbation strength, as it possibly should. In layman's terms, if we only care about a specific number of significant digits and our parameters are non-negative, multiplicative perturbation is a good choice.

However, multiplicative perturbation is somewhat inappropriate for parameters that can be both positive and negative. In this case, it may be desirable to use **additive perturbation** because this will allow crossing zero. In addition, it can also be argued that giving fair weight to every possible parameter value within a parameter's range is advantageous, and this is exactly what additive perturbation accomplishes. In the implementation of this algorithm in OSCILL8, this is a user selectable feature via the `<perturbation_type>` parameter (see Table 3.3).

3.2.2 Adaptive Perturbation: σ and ρ

For the purposes of our discussion here, $\mathcal{U}(c, d)$ indicates a uniform distribution of length $2d$ centered at c and $\mathcal{N}(\mu, \sigma)$ indicates the normal distribution with mean μ and standard deviation σ. We define the **strength of the perturbation**, σ, as an adaptive control (see Figures 3.2(b), 3.4(b) and 3.5(b)) that dictates how far we are currently interested in searching. Then, random samples are either drawn from a uniform distribution ($r_i \in \mathcal{U}(1, \sigma)$ for a multiplicative perturbation and $r_i \in \mathcal{U}(0, d_{p_i} \cdot \sigma)$ for an additive perturbation) or a normal distribution ($r_i \in \mathcal{N}(1, \sigma)$ or $r_i \in \mathcal{N}(0, d_{p_i} \cdot \sigma)$ for multiplicative or additive perturbation, respectively). Here, d_{p_i} is a user-settable characteristic multiplication factor for parameter p_i, which by default is set to $\max p_i - \min p_i$ so that $d_{p_i} \cdot \sigma$ is $100 \cdot \sigma$ % of the total allowable extent of p_i. Then, σ is adapted depending on the **volatility of progress**, ρ, as follows:

$$\sigma \leftarrow \sigma + r \cdot d_\sigma \cdot \text{sign}(\rho), \text{ for a random } r \in [0, |\rho|], \tag{3.7}$$

where, again, d_σ is a characteristic multiplication factor for σ (set by default to be $\max \sigma - \min \sigma$). The volatility of progress, ρ, is essentially a control on the derivative of σ and is itself a function of how the objective function values for the best target structure (that is, progress of the optimization) are improving. There are a number of ways we could imagine controlling this parameter ρ based on history. Currently, we have implemented a simple

Require:
Φ_k, Φ_{k-1}, best fitness from this iteration and the last
I_δ, small difference count from last iteration
I_f, number of iterations to freeze σ adaptation from last iteration

1: *// if progress less than 1/2%, increment I_δ*
2: **if** $\Delta\Phi_k/\Phi_{k-1} < 0.005$ **then**
3: $I_\delta \leftarrow I_\delta + 1$
4: **end if**
5:
6: *// adapt ρ*
7: **if** $I_f > 0$ **then** *// freeze mode*
8: $I_f \leftarrow I_f - 1$
9: $\rho \leftarrow \rho/\phi$
10: **if** $\Delta\Phi_i/\Delta\Phi_{k-1} > 0.005$ **then** *// progress better than 1/2%*
11: $I_f = 3$
12: $\rho \leftarrow \rho/(1 + 10 \cdot \Delta\Phi_k/\Phi_{k-1})$
13: **end if**
14: **else if** $\Delta\Phi_i/\Delta\Phi_{k-1} > 0.005$ **then** *// progress better than 1/2%*
15: $I_f = 3$
16: $\rho \leftarrow \rho/(1 + 10 \cdot \Delta\Phi_k/\Phi_{k-1})$
17: **else if** $\Delta\Phi_i/\Delta\Phi_{k-1} > 0.0005$ **then** *// progress better than 1/20%*
18: $\rho \leftarrow \rho/(1 + 10 \cdot \Delta\Phi_k/\Phi_{k-1})$
19: $I_\delta = 0$
20: **else if** $I_\delta > 5$ **then** *// progress has been bad more than 5 times*
21: $\rho \leftarrow \rho + 0.05 \cdot d_\rho \cdot \text{sign}(\rho)$
22: **else if** $I_\delta > 2$ **then** *// progress has been bad a few times*
23: $\rho \leftarrow \rho + 0.01 \cdot (I_\delta - 1) \cdot d_\rho \cdot \text{sign}(\rho)$
24: **end if**

Figure 3.7: Current algorithm used to adapt ρ. $\phi = \frac{1}{2}\left(1 + \sqrt{5}\right)$ is the golden mean, Φ_k is the *best* current fitness value for the first target population (populations are ordered by preference, with the exception that P_0 is the garbage population), $\Delta\Phi_k = \Phi_k - \Phi_{k-1}$, and again, d_ρ is a characteristic multiplication factor for ρ (set by default to $\max\rho - \min\rho$).

heuristic given in the algorithm in Figure 3.7. More sophisticated versions of ρ-adaptation will be investigated in the future. The variation of ρ for each of the optimization problems T1, T2 and T3 can be seen in Figures 3.2(c), 3.4(c) and 3.5(c).

3.2.3 Full Algorithm

Now, let us flesh out the full algorithm. First of all, there are a fairly large number of parameters used to control the optimization process, which are listed in Tables 3.1 through 3.3 at the end of this chapter. For the purposes of the algorithm, let us define an **intermediate structure**, S_i (with its corresponding **intermediate population**, P_i) as a `<match_diagram>` with $\varepsilon_q = 0$ and a **target structure** (with its **target population**) as a `<match_diagram>` with $\varepsilon_q > 0$ (see Table 3.2, ε_q is the "quit tolerance" saying that we have reached a target to within a fitness value of ε_q). Suppose there are $\xi - 1$ total structures, with $\hat{\xi}$ target and $\xi - 1 - \hat{\xi}$ intermediate. Let P_0 be the **garbage population** of samples that are incompatible with each of the defined S_i. We therefore have a total of ξ sample populations.

Then for a maximum of N_g iterations, we will generate N_s new sample points from each non-empty population P_i based on the algorithm, A. This is done until each target population is non-empty, so that there will be at most $(\xi - 1) \cdot N_s$ samples generated per iteration until we get each target ($\xi - 1$ because there must be at least one target). Once all the target P_i's are non-empty, samples will be divvied up so that each target population generates

$$\hat{\Omega} = \lfloor N_s/(\hat{\xi} + 1) \rfloor + 1 \tag{3.8}$$

samples and all the remaining structures generate

$$\Omega = \left\lfloor (\hat{\Omega} - 1)/(\xi - \hat{\xi}) \right\rfloor + 1 \tag{3.9}$$

samples each (where $\lfloor \cdot \rfloor$ is the "largest integer less than or equal to" operator; i.e., we are doing integer division at each step). In this way, each potential target generates as many samples as all of the remaining non-target populations combined. This is reasonable since

Require:
$\rho \neq 0$
$I_\delta = 0$, incremented when very little or no progress is made

1: *// do N_g generations*
2: **for** $g = 0, \ldots, N_g - 1$ **do**
3: $k \leftarrow$ index of best target population or 1
4:
5: *// adapt ρ and σ*
6: **if** $|P_k| > 0$ **then**
7: adapt ρ, see Figure 3.7
8: **if** original A was 3 **and** $I_\delta > 8$ **then** *// switch A if necessary*
9: $A = 1$ or 2, whichever it is not right now
10: **end if**
11: **end if**
12: **if** $g > 0$ **then**
13: $\sigma \leftarrow \sigma + r \cdot d_\sigma \cdot \text{sign}(\rho)$, for $r \in \mathcal{U}(0, |\rho|)$
14: **end if**
15:
16: *// generate new samples for each valid P_c*
17: **for** $c = 0, \ldots, \xi - 1$ **do**
18: **next** c if $P_c = \varnothing$ **or** (smoldering **and** $P_c \neq$ a target population)
19: **set** number of samples $\zeta = N_s$ or $\hat{\Omega}$ or Ω, see text
20: **for** $a = 0, \ldots, \zeta - 1$ **do**
21: **set** $\nu = a \mod |P_c|$ **if** $A = 1$, or 0 **if** $A = 2$
22: **generate** sample p^* from $P_c[\nu]$, see Figure 3.6
23: **calculate** bifurcation structure $S_{p_\times}(p^*)$ *// **compute intensive***
24: **classify** $S_{p_\times}(p^*) \Rightarrow P_j$
25: **evaluate** $\Phi(S_{p_\times}(p^*), S_j)$
26: **end for**
27: **end for**
28:
29: *// sort and prune*
30: **sort** each P_c by fitness
31: **remove** $|P_c| - N_b$ worst samples so that $|P_c| = N_b$
32: **end for**

Figure 3.8: Main bifurcation matching algorithm. Note that $P_c[a] = a^{th}$ sample from P_c, ordered by fitness. See text for details.

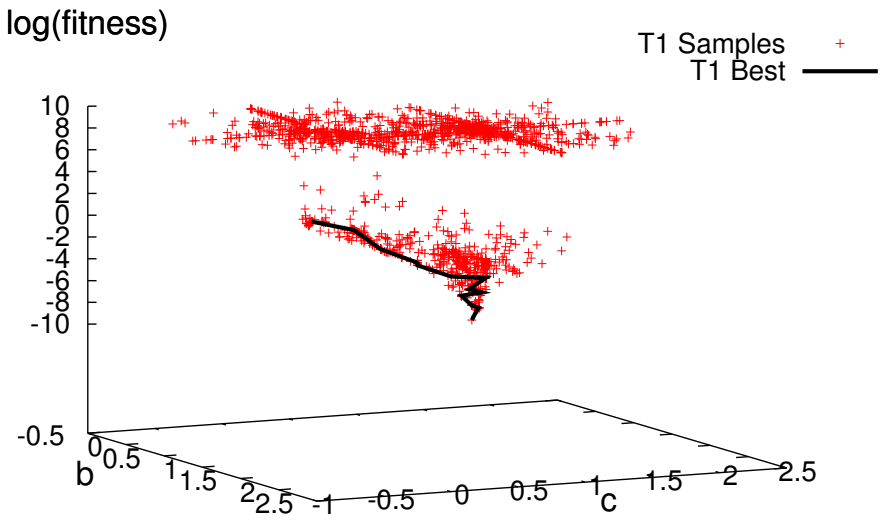


Figure 3.9: Here we can see the fitness landscape for T1 in the parameters b and c. The samples high up are "garbage" samples that were either incompatible with the target or one of the target bifurcation points went out of the valid parameter range (hence, it gets classified as "garbage"). The line follows the best sample to the minimum.

once the targets have been acquired, we should be concentrating most of our effort on them (although leaving other populations active means we have the chance of perturbing yet another, better individual sample into one of the target populations).

The final, full algorithm putting together these details is given in Figure 3.8 along with Figures 3.6 and 3.7.

3.3 Discussion of Results

For our simple model, we ran basic optimizations for compatible and incompatible bifurcation structures. The T1 optimization runs exercise both quantitative (T1) and relative fitness (T1-R) in the context of compatible structure optimization, and also compares results of a slightly restricted search space in T1-P, where p was removed as a valid parameter. T2 adds the complication of incompatible structures, while T3 shows that reversing a particular optimization (recall that T3 optimizes from $S_2 \longrightarrow S_0$, the opposite of T2) may give different results (see discussion below). All optimizations were successful in the sense that they found

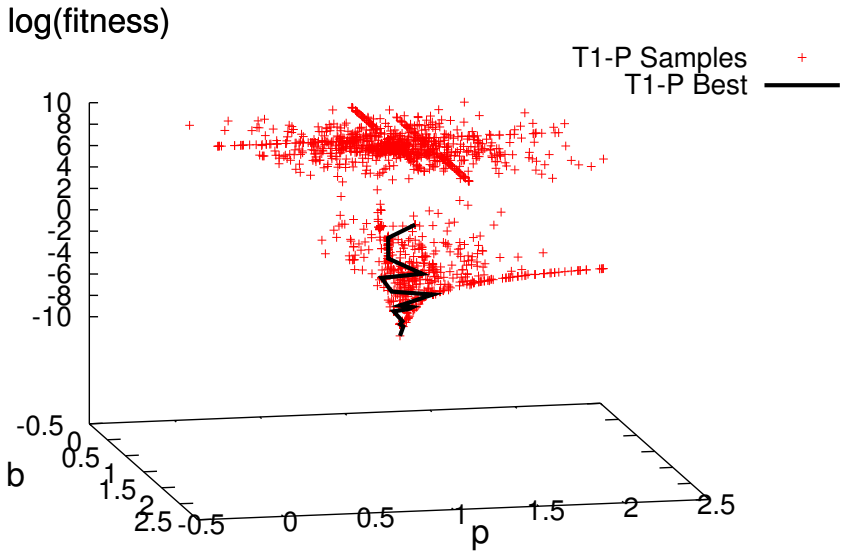


Figure 3.10: Here we can see the fitness landscape for T1 in the parameters b and p. Note in this case that although the parameter p has no bearing on the geometry of the steady state curve, it does play a role in the location of the bifurcation points along that curve, indicated by the changing fitness values in the p direction. We can also see from the structure of these samples that many "garbage" samples p were checked with $p = 1$, the optimal value (indicated by the lines at the top of the sample set).

the exact values for each target structure and they completed in a reasonable number of iterations (not always with the initial choice of algorithmic parameters). Although the simple model is not necessarily representative of the complexity of real biological models (it only has 2 state variables and 4 parameters), it does illustrate the basic mechanisms and modes of the algorithm and gives us some hope that we might be able to apply it to more complex problems (which we do successfully in Chapter 4).

Although the parameter p has no bearing on the geometry of the steady state curve for our simple model (see equation 3.4), it does in fact play a role in the location of the bifurcation points along that curve, indicated by the changing fitness values in the p-direction (Figure 3.10). That is, p is a bona fide player in the optimizations, which means that all of our optimization problems are truly searching 3 degrees of freedom (p, b and c, while a is the parameter on which we are running bifurcation diagrams).

One snag we often run into in performing such optimizations is properly "tuning" the algorithm parameters themselves, which is ironic, since our goal is to "tune" the model!

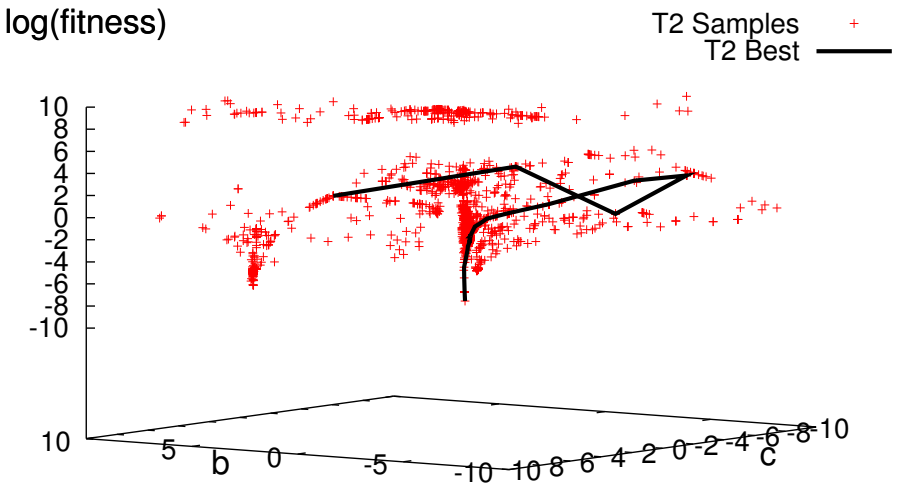


Figure 3.11: Here we can see the fitness landscape for T2 in the parameters b and c. Notice that there are actually 3 different minimums, two of which correspond to the intermediate structures (the one in the middle is the faintest having received only a very few samples). The line follows the best sample into the eventual minimum.

Many times, and this was the case for the first few runs of the T3 optimization, we *fail* to discover the target structure given the parameter settings and the initial parameter ranges. In this case, for each of the T1 and T2 optimizations we used a range of [-10,10] for each parameter, but for the final T3 and T3-R runs, we restricted the ranges of b and c so that $b \in [-5, 5]$ and $c \in [-1, 1]$. Equivalently, this situation could have been remedied by lowering d_b and d_c, the characteristic multiplication constants for these parameters. In general though, each problem may require this type of tuning when progress is not made initially.

Figures 3.9, 3.10, 3.11 and 3.12 give us an idea of how much of the parameter space was searched during each optimization, the fitness landscape, and how the fitness of the best sample improved through that sample space (the solid curve). Although Figures 3.2, 3.4 and 3.5 indicate how the choice of algorithm correlates with progress, we can also graphically study the structure of the sample space in these 3D plots to get the same idea. Indeed, compare the results for T2 (Figure 3.11) and T3 (Figure 3.12), noting the linear structure of searched sample space in T3. This correlates with the 2D plot data (Figure 3.5(a)) which indicates that the single-stepping mode was dominant, whereas the same is not true of T1

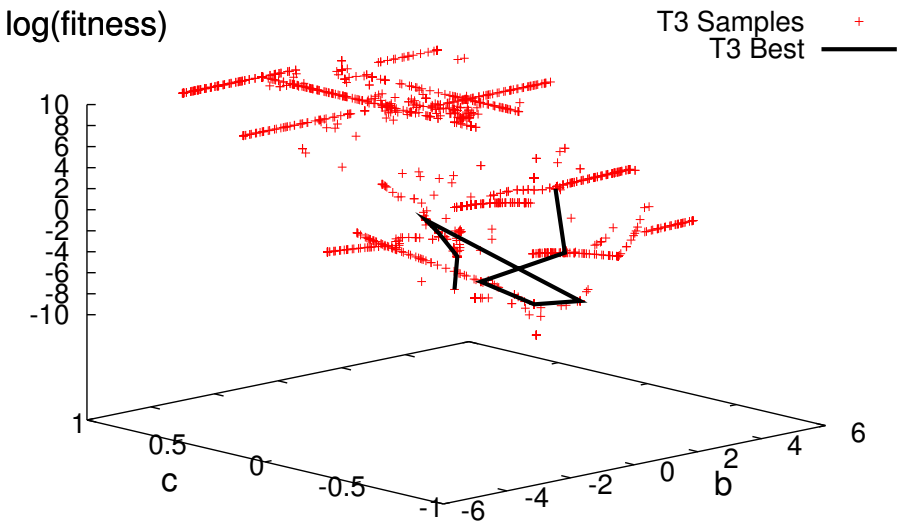


Figure 3.12: Here we can see the fitness landscape for T3 in the parameters b and c. The algorithm almost never switched into $A = 1$ mode, hence the "lines" of progress are clearly visible. This is the single step algorithm.

and T2. The thing that stands out the most in the T2 results (Figure 3.11), however, is that we took a while to find the target structure, evident from the fact that multiple minimums were investigated by the algorithm (for incompatible structures). The upshot is that *both* algorithms are useful and play a role in the optimization scheme.

All in all, I can say that I am pleased with this initial version of the algorithm. There are many improvements that can be made (and will be thoroughly researched), but this algorithm gives modelers a edge in design and exploration that they simply did not have before. As bifurcation structure tends to be relevant in many disciplines that rely on modeling, I see a clear line of research continuing from these initial ideas. Of course, all is not perfect! A number of lessons have been learned along the way, so let us take a moment to detail some of the limitations and future directions for this research.

3.3.1 Limitations and Future directions

There are number of improvements that can be made to the bifurcation matching algorithm, many of which are very simple. They include

- More intelligent σ and ρ adaptation. For simplicity and to get a feel for the mechanism at work, ρ currently changes based only on Φ_k and Φ_{k-1}. An obvious and very likely useful extension would be to modify σ and ρ based on a longer history of the objective function. For example, if you pass by a "sweet spot" of σ (that is, good progress is made for σ^* at some point), keeping this in the memory of the algorithm means that later, when progress is not being made anywhere ($|\rho|$ remains too large and skips all the "sweet spots"), you could return to this value of σ and sit on it for a while. There are an infinite number of ways to adjust the adaptation of σ and ρ and a detailed study of which works best in which situation should probably be done for this algorithm to be more generically useful.

- The current algorithm will hone in on local minima, which is obviously not always the desired mode of operation. A very simple improvement would be to add alternate methods of ordering and pruning each P_i each round. Currently, we use "elitism" by favoring good samples. Instead, we could use a modified "elitism" which attempts to keep only a single best element for each minimum that is being approached (this could be done using statistics on previous samples found in a basin of attraction around a particular minimum, i.e., on a statistical estimation of the contours of the objective function). There are a number of other popular methods used in evolutionary programming that could be applied here as well. This is a good area to research in the future.

- The current algorithm is comparatively weak for compatible structure optimization. That is, it generates more samples that it probably would need to. Although I do not recommend a full-blown steepest decent algorithm such as Levenberg-Marquardt, given the high-dimensional parameter space, a modified form of such an algorithm would be useful. For example, each iteration, i, we do a steepest decent step utilizing (and adjusting) only the parameters $p_i, \ldots, p_{i+9}$ (where the indices are actually taken mod m). In this way, we would limit the number of samples generated per iteration but still make rapid progress. Again, this is a subject which requires further research.

- An obvious limitation of the algorithm is the fact that we use random perturbation

to discover boundaries between incompatible bifurcation structures. A more rigorous method must be investigated.

- A less obvious line of research is to refine the objective function to include more geometry. In the case of one-parameter bifurcation diagrams, we might be interested in the shape of the steady state curve rather than just the layout of the bifurcation points. Even more interesting, maybe we would like to optimize to obtain a two-parameter bifurcation diagram so that geometry would really be necessary. Imagine optimizing a model to expand a domain of oscillation or to get a particular arrangement of bistable and oscillatory domains. This would be very nice.

<run type="BifurcationMatch">

Symbol	Text Name	Default Value	Description
A	<algorithm>	3	algorithm to use, see Table 3.3
T_p	<perturbation_type>	2	perturbation_type to use, see Table 3.3
N_g	<n_generations>	10	maximum number of generations
N_s	<n_samples>	20	basic number of samples
N_b	<n_best>	20	number of samples to keep in each P_i each generation
σ	<sigma>*	<sigma_max>	initial value of σ
σ_{max}	<sigma_max>*	0.1	maximum value of σ
σ_{min}	<sigma_min>*	0.001	minimum value of σ
T_σ	<sigma_type>*	2	*not described in this text*
ρ	<rho>*	<rho_max>	initial value of ρ
ρ_{max}	<rho_max>*	0.1	specifies that $-\rho_{max} < \rho < \rho_{max}$
ρ_{min}	<rho_min>*	0.001	minimum value of $\|\rho\|$
ε_s	<smolder_tol>*	0.0	when $\Phi_k < \varepsilon_s$ for any of P_i with $\varepsilon_q > 0$, we go into smolder mode
$\sigma_{s,max}$	<smolder_sigma_max>*	0.01	σ_{max} used in smolder mode
$\sigma_{s,min}$	<smolder_sigma_min>*	0.001	σ_{min} used in smolder mode
$x_\times$	<sv> or <spe>	0	state variable in diagram
$p_\times$	<par>	0	parameter in diagram
	<match_map>		sub-element, contains several <match_diagram>=S_i elements, see Table 3.2

*Slightly more convenient forms are available for these groups:

- <sigma type="2" min="0.001" max="0.1">0.05</sigma>,
- <rho min="0.001" max="0.1">0.1</rho> and
- <smolder sigma_min="0.001" sigma_max="0.01">0.1</smolder>

Table 3.1: Main control parameters for the optimization algorithm, as implemented in Oscill8. The "Symbol" column refers to the symbol used in this text, the "Text Name" refers to the O8R tag for use with Oscill8's command line and O8R run file syntax and "Default" is the value that will be assumed if not specified by the user. Note that the type of the parameter (i.e., integer, decimal or string) is indicated by the form of the default value.

<match_diagram>

Symbol	Text Name	Default Value	Description
T_d	<type>	0	simple or relative fitness, include x?
γ_s	<scale>	1.0	scale final fitness value (to compare to other diagrams)
γ_p	<penalty>	100.0	penalty applied for incorrect bifurcation point type
ε_q	<eps_q> or <q_tol>	0.0	if Φ of this diagram $< \varepsilon_q$, this diagram is optimized (see ε_s for significance to smolder mode)
	<match_point>		sub-element, see below

<match_point>

Symbol	Text Name	Default Value	Description
$\hat{x}_\times$	<x_value>	0.0	value of <sv> to target
$\hat{p}_\times$	<p_value>	0.0	value of <par> to target
τ	<tau> or <type>	0	bifurcation type
ω	<omega> or <weight>	1.0	relative weight this point should have in diagram

Table 3.2: <match_diagram> and <match_point> control parameters, as implemented in OSCILL8. See Table 3.1 for main control parameters.

<algorithm>

1	vector perturbations, many
2	single perturbations, best
3	adaptive, both vector and single

<perturbation_type>

1	multiplicative perturbation
2	additive perturbation

Table 3.3: Settings for the <algorithm> and perturbation_type parameters. Currently, there are only two different algorithms that differ primarily in the way in which samples are generated. In addition, option 3 triggers switching from one algorithm to the other when sufficient progress is not being made with one of the algorithms. Multiplicative perturbation explores different magnitudes of a parameter equally while additive perturbation aims to give equal weight to all possible values of a parameter in some fixed range.

Chapter 4

Circadian Models

Now we will develop several different models of the circadian clock in *D. melanogaster*, starting with a modified form of Goldbeter's 1995 model (model M0). To this simple model, we will add the effects of a rhythmically expressed phosphatase (models M1 and M2), *tim* mRNA and protein (model M3) and finally *vri*, *pdp1ε*, *clk* and *cry* (model M4). This will take us from a model of 4 state variables, 17 parameters (for M0) all the way up to 24 state variables, 75 parameters (for M4). The intent is to show how the computational tools (Chapter 2) and bifurcation analytic techniques (Chapter 3) presented in this dissertation can help a modeler add complexity and test interesting hypotheses in an expedient manner. In addition, we will explore recent experimental results and discuss the biological significance these models can help to shed light on.

4.1 M0

We will take as a starting point a basic four component system which is a simplification of the model of Goldbeter (1995). The difference between this model and his is that we have removed the bisphosphorylated form of PER (an extra element causing delay in the negative feedback loop) and included addition background degradation. Figure 4.1 gives a wiring diagram of the hypothesis. This model has the following associated set of nonlinear ODEs:

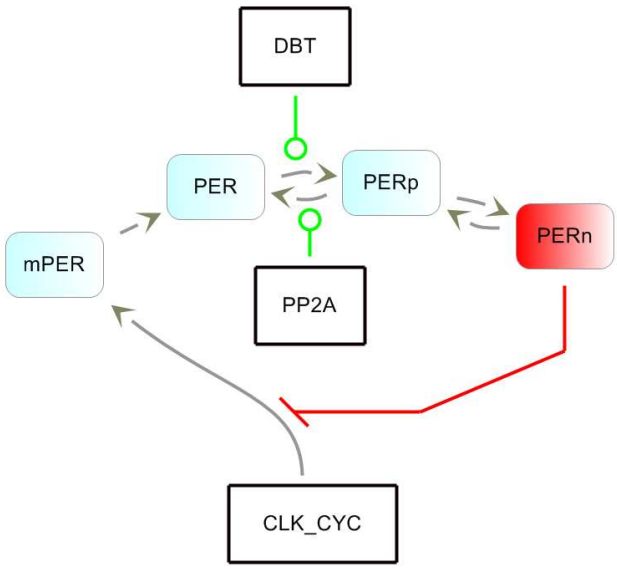


Figure 4.1: Our base model, M0. This is a simplified form of Goldbeter's 1995 model Goldbeter (1995), removing the bisphosphorylated form of PER and adding background degradation. Note that CLK-CYC, DBT (δ) and PP2A (π) are constants here. This model has 4 state variables and 17 parameters (two are redundant, see text). See equations (4.1) for the associated ODEs.

$$\frac{dM_P}{dt} = b_0 + \frac{V_0}{1 + (P_n/K_i)^n} - d_0 M_P \tag{4.1a}$$

$$\frac{dP_0}{dt} = s_1 M_P - \delta \frac{V_\delta P_0}{J_\delta + P_0} + \pi \frac{V_\pi P_1}{J_\pi + P_1} - d_1 P_0 \tag{4.1b}$$

$$\frac{dP_1}{dt} = \delta \frac{V_\delta P_0}{J_\delta + P_0} - \pi \frac{V_\pi P_1}{J_\pi + P_1} - k_{in} P_1 + k_{out} P_n - d_2 P_1 \tag{4.1c}$$

$$\frac{dP_n}{dt} = k_{in} P_1 - k_{out} P_n - d_3 P_n \tag{4.1d}$$

We have four state variables representing the concentrations of *per* mRNA (M_P), PER (P_0), phosphorylated PER (P_1), and a nuclear, inhibitory form of phosphorylated PER (P_n). Let us first note that this system is constructed with a basic negative feedback loop at its core:

$$\text{mPER} \rightarrow \text{PER} \rightarrow \text{PER-P} \rightarrow \text{PER}_n \dashv \text{mPER},$$

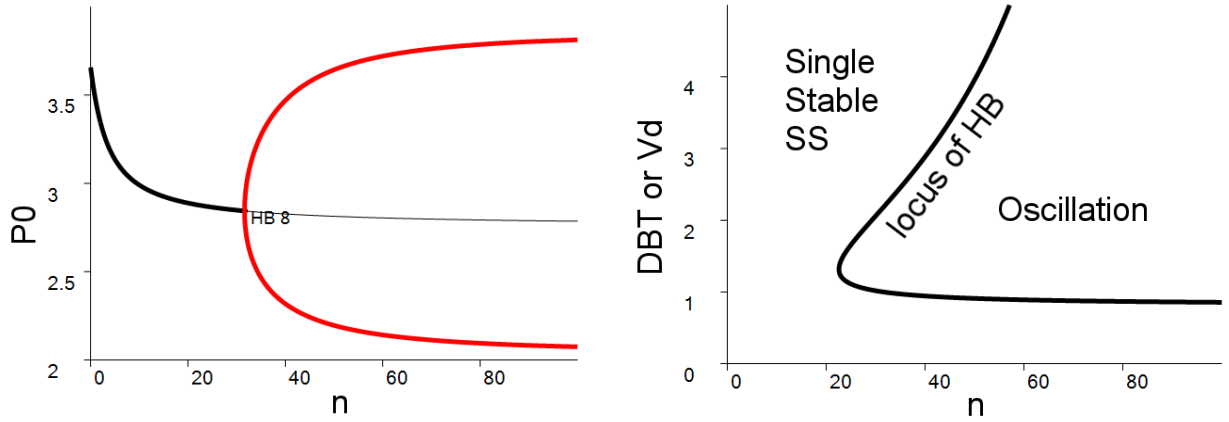
which implements the idea that PER inhibits its own transcription. Second, we have explicitly indicated that DBT (δ) and PP2A (π) are responsible for the phosphorylation and dephosphorylation of PER, though in this first version of the model, they are parameters rather than dynamic variables (note that δ and π are redundant in the equations; it is only the products δV_δ and πV_π that have significance). The same is true of CLK-CYC,

Name	Init	T1*	T1	T2*	T2	Init - T2
n	2	–	–	–	–	0
δ	1*	–	–	–	–	0
π	1*	–	–	–	–	0
J_δ	1	3.48388	0.1*	0.183677	0.2	0.8
J_π	1	0.001	0.01*	–	0.01	0.99
K_i	1	1.41455	0.1*	0.019785	0.01	0.99
d_0	0.1	–	0.5	0.2	0.2	-0.1
d_1	0.1	0	0.25	3.44083	0.4	-0.3
d_2	0.1	0	0.25	0	0.1	0
d_3	0.1	–	0.25	0.15	0.2	-0.1
k_{in}	1	0.108408	0.2*	0.121796	0.05	0.95
k_{out}	1	0.01	0.05*	0.0386	0.05	0.95
b_0	0.1	0	0	0	0	0.1
s_1	0.1	2.42771	1	1.39772	0.4	-0.3
V_δ	1	1.0527	0.75	1.32125	1	0
V_π	1	0.001	0.4	0.571	0.5	0.5
V_0	1	2.33646	1	0.9696633	5	-4

Table 4.1: M0 initial and optimized parameter values. The initial parameter set cannot sustain oscillation except for very large values of n (i.e., > 22). The optimized sets were obtained using the techniques of Chapter 3 to get oscillation for $n \approx 1$. Each of T1 and T2 correspond to a single optimization run (taking just a few minutes) followed by twiddling (for T1, about 5 minutes, for T2, about 1 hour)). Columns T1* T2* have the actual values output from the T1 and T2 optimizations for comparison. When a value is marked with * in one column, this indicates that the following optimization had this value either fixed or in a fairly restricted range. Those marked with – are the same as the previous column.

although CLK-CYC is not represented explicitly as a constant in this model. Later, as we add complexity to the model, some of these species will become dynamic variables.

The first thing we need to do is to find a region of oscillation. Recalling that negative feedback oscillators typically produce oscillation via a Hopf bifurcation, which happens due to the time delay experienced in the loop, we can focus on the parameters that produce this effect. There are a number of ways to do this, but one way which includes adjusting only one parameter is to increase the cooperativity of the inhibition of PER_n on transcription of mPER, represented by the Hill exponent, n. Given the initial parameter settings in Table 4.1, we generate a one parameter bifurcation diagram on the parameter n, for $n \in [0, 100]$. Lo and behold, we find a HB at $n \approx 32$ (see Figure 4.2(a)). But $n \approx 32$ is NOT an acceptable



(a) PER vs. n (b) DBT vs. n

Figure 4.2: a) One parameter bifurcation diagram of $n \in [0, 100]$. The maximum and minimum of the oscillatory solution is also depicted (in red) to the right of the HB. b) Two parameter bifurcation diagram of DBT or V_d vs. n. The region labeled as "Oscillation" is where we have limit cycle oscillations. Note that the minimum n we can choose and still have oscillation (for this parameter set, corresponding to the initial parameter set in Table 4.1) occurs for $n \approx 22$, with DBT or $V_d \approx 1.3$.

value for n, since cooperativity coefficients are more realistically less than 3 or 4. Now we can begin to do many two-parameter continuations of the HB with respect to other parameters in the system searching for parameter combinations that will lower the value of n at the HB. We may also slowly "tweak" parameters and quickly update the one-parameter bifurcation diagram on n. In the past, this was almost an intolerable way to proceed given the effort involved in generating one- and two-parameter bifurcation diagrams. OSCILL8 has addressed this particular hassle in a few ways:

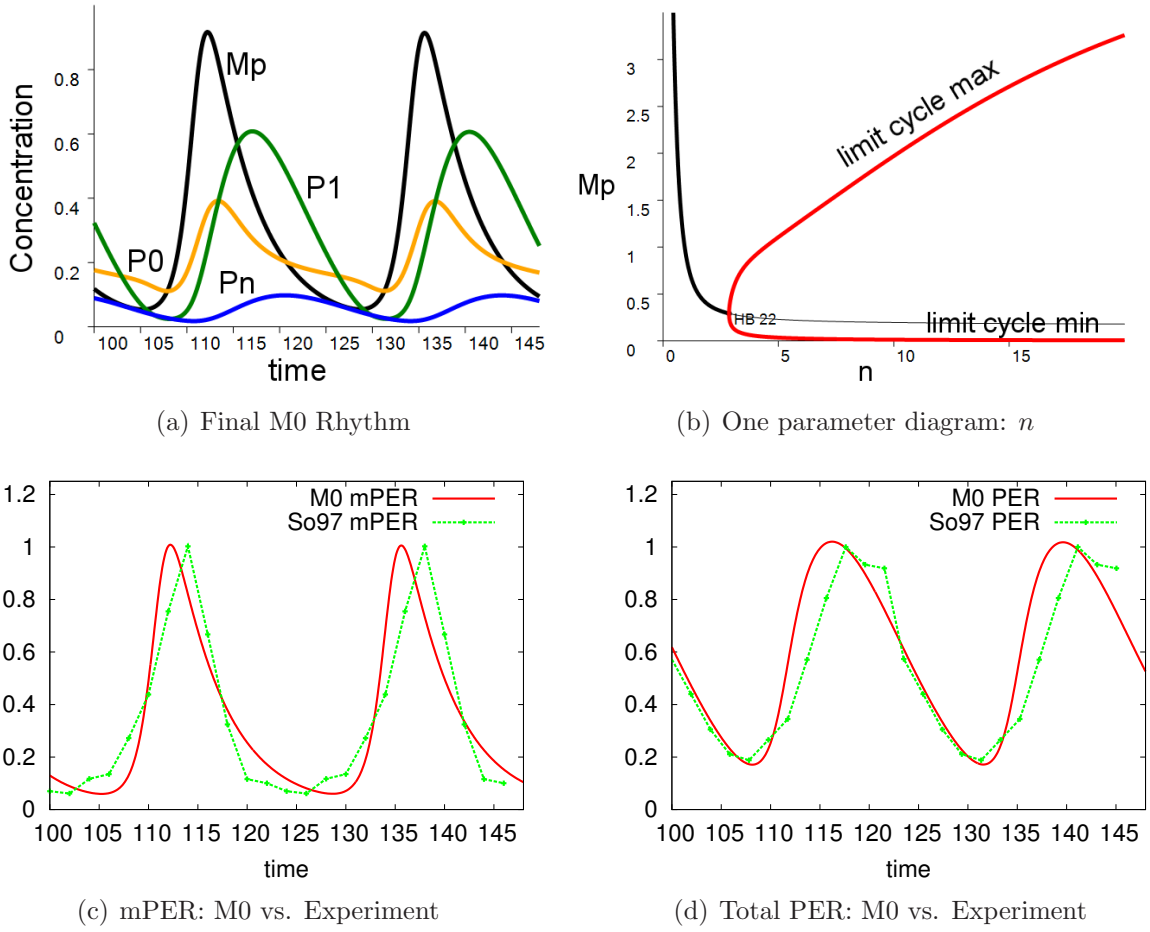
- OSCILL8 allows batched two-parameter continuation (see Chapter 2). On my P4, 2.4GHz, 1 GB RAM laptop it takes about 30 seconds to generate all 16 possible two-parameter bifurcation diagrams vs. n. One such two-parameter diagram is shown in Figure 4.2(b), where we see that increasing n past $(n, V_d) \approx (22, 1.3)$ across the locus of Hopf bifurcation leads us into the region of oscillation; certainly $n = 22$ is better, but still not great. We can adjust parameters to be just inside this domain of oscillation for $n \approx 22$ and repeat the process (of generating two-parameter bifurcation diagrams). For example, reducing $b_0 = 0$, we get a HB for $n \approx 18$. Following other leads in the two-parameter bifurcation diagrams for d_0 and s_1, we quickly reduce the HB down to around $n \approx 10$, but then it appears that we will have to start changing combinations

of parameters in order to make progress (all two-parameter bifurcation diagrams have minima at $n \approx 10$, indicating many minute adjustments must be made in order to make any progress). At this point, even this particular method leads to fatigue and frustration!

- OSCILL8 also has the "parameter twiddler" interface, which is in some circumstances an even quicker way to adjust parameters (see Chapter 2). Using this feature (in combination with batched two-parameter continuation), it took me about 10 minutes to twiddle parameters to get the HB down around $n \approx 10$. On the other hand, this also becomes tiring, since at some point, intuition begins to fail you and it is hard to predict exactly what combination of of 16 different parameters will achieve the results we want.

Once you have tried these methods a few times, you get the hang of it and can make good progress up to a point. In the end though (especially as we get closer to our goal), some automated adjusting scheme becomes *very* useful (and in most cases, we will abandon any other techniques very quickly in favor of these automated ones). Using OSCILL8's bifurcation matching algorithm (see Chapter 3), we can automatically search for a parameter set that lowers the value of n at the HB.

We run a couple of consecutive optimizations (labeled T1 and T2) trying to lower the HB to occur at $n = 1$ (for the complete O8R files with the parameters used to run these optimizations, see Appendix B). Each optimization run was followed by a bit of hand twiddling to adjust the waveforms and to get certain parameters to more reasonable values. A perfect example of the need to twiddle following an optimization would be the discovered values of J_δ vs. J_π and V_δ vs. V_π in optimization T1. Logically, we might expect each of these two parameter groups to be roughly of the same order, given that they are the constants controlling opposing kinase and phosphatase activity. Unfortunately, the optimization T1 found that pushing J_δ up to 3.5 with J_π down to 0.001 (which was the minimum value allowed for both J_δ and J_π) and V_π down to 0.001 (again, the minimum allowed value) was one of the best, steepest routes to get the value of n at the HB down closer to 1. So, following the optimization, we try to bring J_δ down and J_π and V_π up a bit at a time, by hand, while



(a) Final M0 Rhythm

(b) One parameter diagram: n

(c) mPER: M0 vs. Experiment

(d) Total PER: M0 vs. Experiment

Figure 4.3: After several rounds of optimization, we find that we can reproduce circadian behavior quite well. a) Although oscillation is possible for $n = 2$ for a very nearby parameter set, we have displayed here $n = 4$ (to correspond to the behavior of Goldbeter (1995)). In b), we can see the final, optimized version of the bifurcation structure for n, allowing oscillation for much lower values of n (compare with Figure 4.2(a)). In c) and d), we see that although the exact shape of the wave forms are not exact, they are not terribly far from quantitatively correct.

slightly adjusting parameters. Then, for each subsequent optimization run we can fix the parameters that we are happy with and restrict the valid range for other parameters making our search more and more specific. In Table 4.1, we have included the actual values at the end of the T1 optimization (before twiddling) in the column marked T1* to indicate what was done by hand after the optimization, which in this case was quite a lot (column T1). In fact, given the strange values for J_δ, J_π, V_δ and V_π, optimization T1 was stopped early to adjust parameters and begin with better constraints (this is often the easiest way to make fast progress).

Algorithm Improvement:
We should point out here that one significant enhancement to the optimization algorithm that would help us to avoid the repeated optimization/twiddle sequence just described would be the addition of generic constraints on parameters. These would look like, e.g., $p_1/4 < p_2 < 10p_1$ or $p_1 = 3p_2$. These could be added to the objective function as additional criteria in a straight forward manner and would be a nice addition to the algorithm.

In the end, after two rounds of optimization, we get M0 to behave with realistic circadian behavior (see Figure 4.3). Although oscillation is possible with $n = 2$ for basically the same parameter set, we choose here to display circadian behavior for $n = 4$. Also, because of the known effects of DBT on the stability of PER (i.e., that is becomes more unstable), one might argue that we should have $d_2 > d_1$. This is probably correct, but unimportant for such a simple model. In the next section, we will be sure to get a more reasonable value for d_2 with respect to other degradation constants since the next model concentrates on kinase and phosphatase activity.

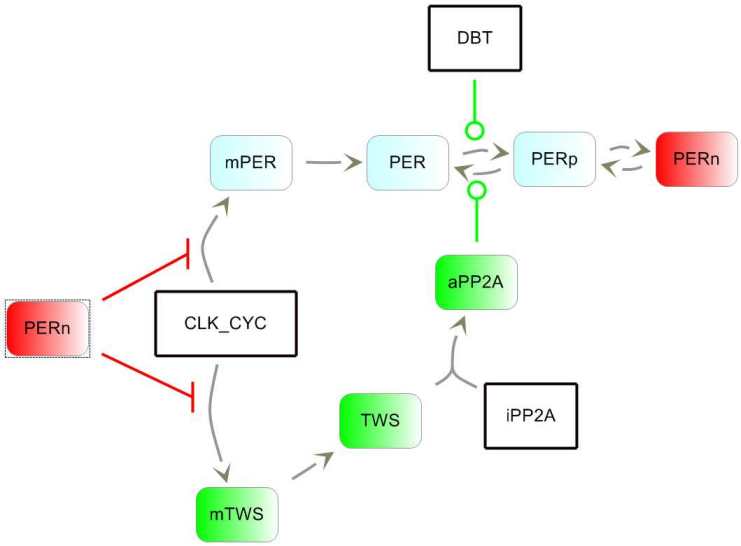


Figure 4.4: $M1$. This takes model M0 up a notch, to include detail related to the regulation of PP2A via the gene *tws*. Note that *tws* is regulated by CLK-CYC, a hypothesis made by Sathyanarayanan et al. (2004). This model has 7 state variables and 23 parameters. See equations (4.2) for the ODEs used to analyze this hypothesis.

4.2 M1

The next level of complication we would like to add is the gene *tws*, which codes for the regulatory B-subunit of PP2A, making PP2A active as a phosphatase on PER (see Section 1.1 for more details on this gene). *tws* mRNA (M_{T_π}) is translated into TWS (T_π) which combines with inactive PP2A (π_i) to make active PP2A (π_a). Figure 4.4 shows the basic wiring diagram of the model. This model captures the interesting hypothesis made by Sathyanarayanan et al. (2004) that *tws* is regulated via CLK-CYC. The model equations (also found as an ODE file

in Appendix A) are:

$$\frac{dM_P}{dt} = b_0 + \frac{V_0}{1+(P_n/K_i)^n} - d_0 M_P \tag{4.2a}$$

$$\frac{dP_0}{dt} = s_1 M_P - \delta \frac{V_\delta P_0}{J_\delta + P_0} + \pi_a \frac{V_\pi P_1}{J_\pi + P_1} - d_1 P_0 \tag{4.2b}$$

$$\frac{dP_1}{dt} = \delta \frac{V_\delta P_0}{J_\delta + P_0} - \pi_a \frac{V_\pi P_1}{J_\pi + P_1} - k_{in} P_1 + k_{out} P_n - d_2 P_1 \tag{4.2c}$$

$$\frac{dP_n}{dt} = k_{in} P_1 - k_{out} P_n - d_3 P_n \tag{4.2d}$$

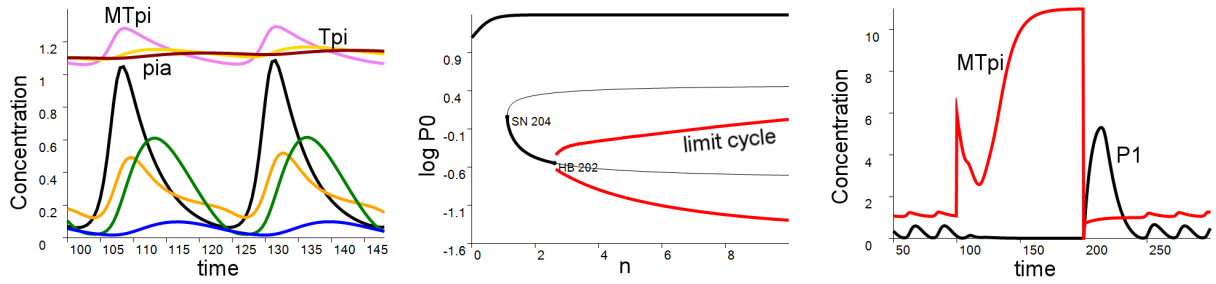
$$\frac{dM_{T_\pi}}{dt} = b_4 + \frac{V_4}{1+(P_n/K_i)^n} - d_4 M_{T_\pi} \tag{4.2e}$$

$$\frac{dT_\pi}{dt} = s_5 M_{T_\pi} - k_5 \pi_i T_\pi + k_{-5} \pi_a - d_5 T_\pi \tag{4.2f}$$

$$\frac{d\pi_a}{dt} = k_5 \pi_i T_\pi - k_{-5} \pi_a \tag{4.2g}$$

Notice that equations (4.2a)–(4.2d) describing the kinetics of *per* and its progeny are identical to model M0, giving us a 4 variable system which we have already worked on and have, therefore, parameter estimates to start with. Equations (4.2e)–(4.2g) describe the dynamics of *tws* products and PP2A, giving us 3 more variables, for a total of 7 state variables. In addition to the 16 parameters from M0 (the parameter π from M0 is now a state variable, π_a, so we only inherit 16 parameters), we add 8 new parameters for a total of 24 parameters (one of the new parameters, π_i is again redundant since it only appears in the product $k_5\pi_i$). Parameter values for M1 can be found in Table 4.2.

The first fun thing to notice is that equations (4.2a)–(4.2d) are coupled to equations (4.2e)–(4.2g) via *only* π_a (which replaces the role of the constant π from the previous model) so that if $\pi_a \approx 1$, equations (4.2a)–(4.2d) should behave exactly as they did in M0. Indeed, using the initial parameter set from M0 and the reasonable guesses for the 8 new parameters (see Table 4.2), we get Figure 4.5. Although the waveforms are not correct, we can certainly work to adjust them so they are. More interestingly though, is the bifurcation diagram for n. It looks quite a bit different than our previous model! In particular, we notice that there is now a region of bistability for $1 < n < 2.6$ (with two alternate stable steady states) and for $n > 2.6$ (with a stable steady state and a stable limit cycle). This is



(a) Initial TS (b) Initial 1P BD (c) Initial Jump Behavior

Figure 4.5: a) M1 initial wave forms, illustrating how much better of we start with our analysis of M1 than we did with M0. Compare with Figure 4.3. b) On the other hand, our bifurcation structure is VERY different! It allows temporal changes in state variables to cause the system to jump between oscillatory behavior and equilibrium. c) At $t = 100$, $M_{T_\pi} \rightarrow 6$, causing a jump from oscillatory to equilibrium behavior, and at $t = 200$, $M_{T_\pi} = T_\pi = \pi_a = 0$, causing a jump back to oscillatory behavior. This ability to jump back and forth is due to the positive feedback in the system.

a very interesting dynamical feature, indeed, because it introduces the possibility of making temporal changes to state variables (as opposed to parameters, which can always cause major dynamic changes) to move the system between oscillatory and equilibrium behavior (see Figure 4.5(c)).

Furthermore, the SN bifurcation and region of bistability are results of the positive feedback loop mTWS $\rightarrow$ TWS $\rightarrow$ PP2A $\dashv$ PER-P, thereby releasing the inhibition PER-P $\dashv$ mTWS via the transcription factor CLK-CYC. The jump behavior described above is a natural consequence of how our model is wired together depending on positive and negative feedback. On the one hand, we have a stabilizing negative feedback loop that will drive the system toward equilibrium unless we increase the delay in the loop enough to push the system into a limit cycle oscillation. Nevertheless, this limit cycle is stable and the system will relax into it similar to the way it settles into a stable steady state. On the other hand, we have an antagonistic positive feedback loop that pushes the system toward instability. As we increase the rate of any of the reactions involved with *tws* (or even "dump" extra amounts of one of its products into the system–i.e., increase one of M_{T_π}, T_π or π_a at some point in time), the system will tend to destabilize, as in Figure 4.5(c) where at $t = 100$, we move $M_{T_\pi} \rightarrow 6$, causing a jump from oscillatory to equilibrium behavior. Then, again, at $t = 200$, we set $M_{T_\pi} = T_\pi = \pi_a = 0$, causing a jump back to oscillatory behavior.

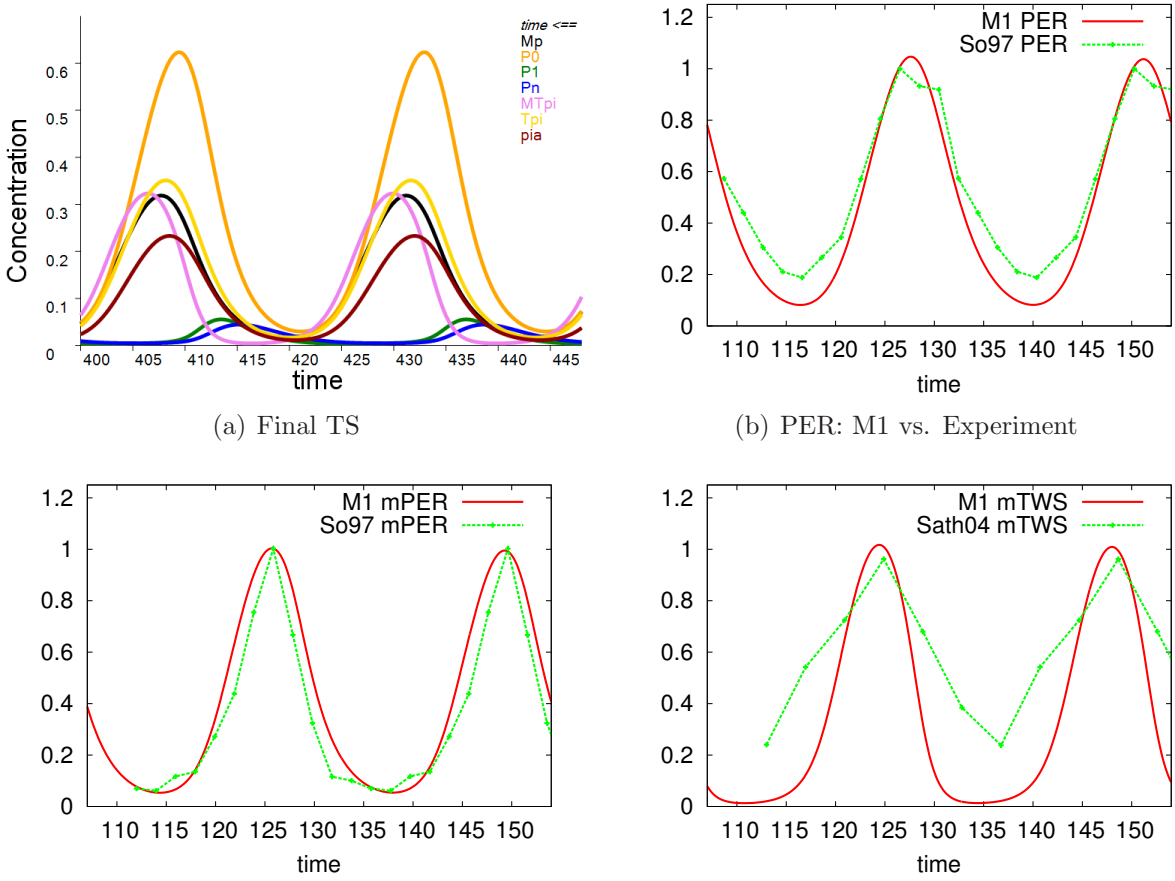
Now, we would like to get the model to match, at least qualitatively, the evidence

Name	Init	Final	Δ	Name	Init	Final	Δ
n	4	2	2	b_0	0	–	0
J_δ	0.2	2	-1.8	s_1	0.4	0.87	-0.47
J_π	0.01	0.001	0.009	b_4	0.1	0	0.1
K_i	0.01	0.001	0.009	s_5	0.1	0.90625	-0.80625
d_0	0.2	0.3625*	-0.1625	V_δ	1	1.99375	-0.99375
d_1	0.4	0.3625*	0.0375	V_π	0.5	2.175	-1.675
d_2	0.1	0.90625	-0.80625	V_0	5	3.2625	1.7375
d_3	0.2	0.3625*	-0.1625	V_4	1	9.96875	-8.96875
d_4	0.1	1.26875	-1.16875	π_i	1	–	0
d_5	0.1	0.725	-0.625	k_5	0.1	1.8125	-1.7125
k_{in}	0.05	0.453125	-0.403125	k_m5	0.1	2.71875	-2.61875
k_{out}	0.05	0.090625	-0.40625	δ	1	–	0

Table 4.2: M1 initial and final parameter values. Final values reflect a scaling up in order to make a $\approx$ 24 hr. oscillation. Those marked with a * were in fact the same before as the initial before the 81.25% inflation.

from Sathyanarayanan et al. (2004). Specifically, they have noted that mTWS (the so called "P1" transcript) oscillates with a 5-fold amplitude (and give times series data), and although they did not report any levels for the TWS protein (by itself or in complex with PP2A), we will assume that this protein (and hence the active form of PP2A, π_a) in fact has a rhythm (although this is not always the case, e.g., *cry* mRNA cycles though the protein does not). We will not be picky whether it is 3-fold or 5-fold. They also note that overexpression of *tws* results in arrhythmia (which we may associate with a steady state of the model). As our final goals, we would like to lower $n \to 2$ since the addition of the positive feedback gives us much more dynamical flexibility and ensure that the effects of DBT are clear (i.e., d_2 is larger than other related degradation constants).

Using the initial set above, we see that M_{T_π} does indeed have a rhythm, but it is not even 2-fold. In fact, we would like the rhythm of M_{T_π} to look similar to that of M_P given that they are similarly regulated. This is a matter of equalizing the parameters that control the transcripts, namely b_4, V_4 and d_4, while maintaining the overall rhythm. One quick way to bring down the overall levels of the *tws* products is to recall that $\pi_a V_p = 1$ (for the initial set and in M0, $\pi_a = 1$ and $V_p = 0.5$), so that swapping these values to get $\pi_a = 0.5$ and $V_p = 1$ allows oscillation for a much lower level of phosphatase activity. Then we begin to



(a) Final TS

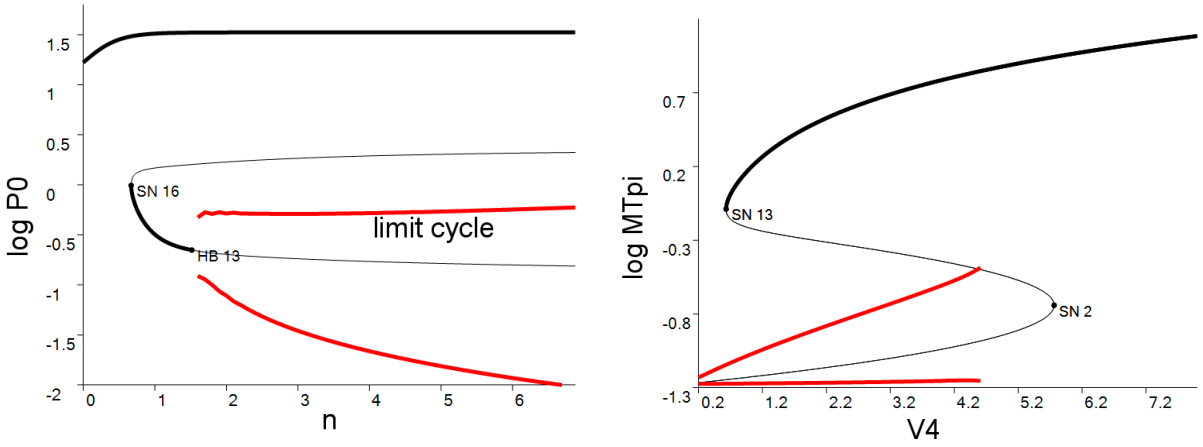
(b) PER: M1 vs. Experiment

(c) mPER: M1 vs. Experiment

(d) mTWS: M1 vs. Experiment

Figure 4.6: After some optimization and hand twiddling, we get good approximation to quantitative details of the mPER (c) and total PER rhythms (b) in model M1. On the other hand, while mTWS is qualitatively correct with respect to its roughly 5-fold amplitude (d), the character of the solution for mTWS is clearly wrong. With optimization tools geared toward quantitative fitting, we could presumably resolve this issue. Note that data is scaled to get roughly matching levels. Note in (a) that $n = 2$ and in the legend "MTpi", "Tpi" and "pia" stand for M_{T_π}, T_π and π_a, respectively.

modify b_4, V_4 and d_4 in conjunction with the degradations constants and the T_π constants to fit the model's rhythm. Although it is hard not to wish for an automated method for fitting time series data, using the bifurcation matching algorithm in conjunction with the the parameter twiddler feature of OSCILL8, we can rapidly achieve the necessary results. We will not again detail the use of the matching algorithm, but suffice it to say that after two rounds of optimization, we achieved results satisfactory enough to continue on with twiddling.



(a) Bifurcation Diagram for n

(b) Bifurcation Diagram for V_4

Figure 4.7: In (a) we see that there is a HB for $n < 2$, making oscillation possible for $n = 2$. The associated time series for $n = 2$ can be seen in Figure 4.6(a). In (b), we see that oscillation is possible for $V_4 < 4.7$ (with a default value of 4) and that increasing V_4 past 4.7 brings us to a high M_{T_π} steady state after colliding into an unstable steady state. (These figures do not correspond to the final parameter set—note that $V_4 = 4$ is the default value in this picture—but are similar to, though less complicated than, the actual final bifurcation structure. They are meant to be illustrative of the type of bifurcation structure we have access to in this model.)

4.2.1 Discussion of M1 Results

In Figure 4.6 and 4.7, we see that our basic goals have been achieved. While maintaining the normal *per* rhythm at $n = 2$ (Figures 4.6(b) and 4.6(c), with $n = 2$ from Figure 4.7(a)), Figure 4.6(d) shows that mTWS has a (more than) 5-fold oscillation (although obviously the quantitative details are not quite correct). Furthermore, Figure 4.7(b) shows that overexpression of *tws* by a factor of 2 leads to arrhythmia (steady state) via a saddle-loop bifurcation (although this diagram does not correspond exactly to the final parameter set, the actual final bifurcation structure has the same property). Further refinement of the parameter values is possible using optimization techniques not presented in this thesis.

To complete our modeling of known phosphatase activity in *D. melanogaster*, we should include *wdb*, another gene with oscillating transcript whose protein product is second regulatory B-subunit for PP2A. One confounding aspect of the evidence reported by Sathyanarayanan et al. (2004) is that, although mWDB cycles with a circadian period, it cycles in anti-phase to mTWS, indicating a different regulatory mechanism. We might therefore speculate that *wdb* is regulated in a manner similar to *clk* (see Figure 1.1(b), lower panel for dClk rhythm), but as of yet there is no evidence to support this hypothesis. Another

interesting piece of evidence is that WDB tends to be nuclear unlike TWS, which is typically localized in the cytoplasm. This may indicate different roles for WDB and TWS, whereby TWS stabilizes PER in the cytoplasm during the night and WDB stabilizes PER in the nucleus during the early day so that it may properly inhibit CLK-CYC activity. We will leave this modeling task for another time when there is a bit more experimental evidence to support various hypotheses.

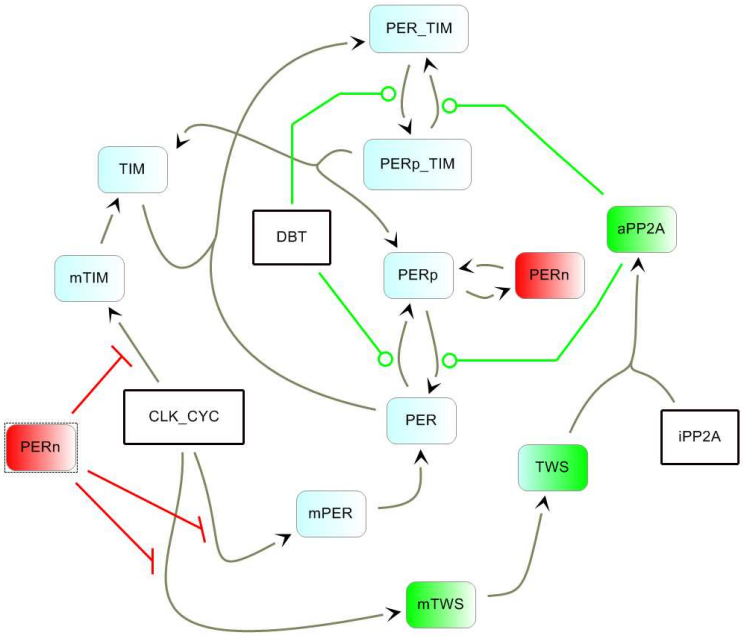


Figure 4.8: M3. This takes model M1 up a notch to include detail related to the *tim* gene. TIM can bind PER, thereby stabilizing it. This model has 11 state variables and 39 parameters. See equations (4.3), (4.4) and (4.5) for the ODEs used to analyze this hypothesis.

4.3 M3

The next level of complication we would like to add is the gene *tim*, which codes for PER's binding partner TIM (see Section 1.1 for details). TIM apparently serves two main functions. The first is to stabilize PER (by binding), making TIM and TWS partners in the struggle to keep PER around long enough to perform the requisite inhibition of its own gene. The second is to provide a means for light-input to the clock, which happens when a light-activated conformational change in CRY allows it to bind TIM, marking it for swift degradation by the proteosome. Originally, it was thought that TIM was a requisite binding partner of PER in order that the pair could enter into the nucleus. It now appears this is not the case since PER can be found inside the nucleus well before TIM (Shafer et al., 2002). Light-input will be explored in the next model, M4. The model equations for M3 (also found as an ODE file

in Appendix A) are:

$$\frac{dM_P}{dt} = b_0 + \frac{V_0}{1+(P_n/K_i)^n} - d_0 M_P \tag{4.3a}$$
$$\frac{dP_0}{dt} = s_1 M_P - \delta \frac{V_\delta P_0}{J_\delta + P_0} + \pi_a \frac{V_\pi P_1}{J_\pi + P_1} - k_8 P_0 T_0 + k_{-8} C_0 - d_1 P_0 \tag{4.3b}$$
$$\frac{dP_1}{dt} = \delta \frac{V_\delta P_0}{J_\delta + P_0} - \pi_a \frac{V_\pi P_1}{J_\pi + P_1} - k_{10} P_1 T_0 + k_{-10} C_1 - k_{in} P_1 + k_{out} P_n - d_2 P_1 \tag{4.3c}$$
$$\frac{dP_n}{dt} = k_{in} P_1 - k_{out} P_n - d_3 P_n \tag{4.3d}$$

$$\frac{dM_{T_\pi}}{dt} = b_4 + \frac{V_4}{1+(P_n/K_i)^n} - d_4 M_{T_\pi} \tag{4.4a}$$
$$\frac{dT_\pi}{dt} = s_5 M_{T_\pi} - k_5 \pi_i T_\pi + k_{-5} \pi_a - d_5 T_\pi \tag{4.4b}$$
$$\frac{d\pi_a}{dt} = k_5 \pi_i T_\pi - k_{-5} \pi_a \tag{4.4c}$$

$$\frac{dM_T}{dt} = b_7 + \frac{V_7}{1+(P_n/K_i)^n} - d_7 M_T \tag{4.5a}$$
$$\frac{dT_0}{dt} = s_8 M_T - k_8 P_0 T_0 + k_{-8} C_0 - k_{10} P_1 T_0 + k_{-10} C_1 - d_8 T_0 \tag{4.5b}$$
$$\frac{dC_0}{dt} = k_8 P_0 T_0 - k_{-8} C_0 - \delta \frac{V_{\delta 2} C_0}{J_{\delta 2} + C_0} + \pi_a \frac{V_{\pi 2} C_1}{J_{\pi 2} + C_1} - d_9 C_0 \tag{4.5c}$$
$$\frac{dC_1}{dt} = \delta \frac{V_{\delta 2} C_0}{J_{\delta 2} + C_0} - \pi_a \frac{V_{\pi 2} C_1}{J_{\pi 2} + C_1} + k_{10} P_1 T_0 - k_{-10} C_1 - d_{10} C_1 \tag{4.5d}$$

The first 7 equations are similar to M1, adding only the effects of binding into and releasing from the complexes C_0 and C_1 (in the equations for P_0 and P_1, respectively). *tim* mRNA (M_T) is translated into TIM protein (T_0) and subsequently added to the complexes C_0 and C_1, which represent the PER-TIM dimer in which PER can either be phosphorylated (C_1) or not (C_0). The addition of TIM basically gives PER a holding ground where it is relatively stable compared to its unstable monomeric and phosphorylated form, P_1. This gives PER a chance to build up so that it may perform its required duties. Note that the total amount of PER and TIM (each non-conserved) are

$$P_t = P_0 + P_1 + P_n + C_0 + C_1 \text{ and}$$
$$T_t = T_0 + C_0 + C_1.$$

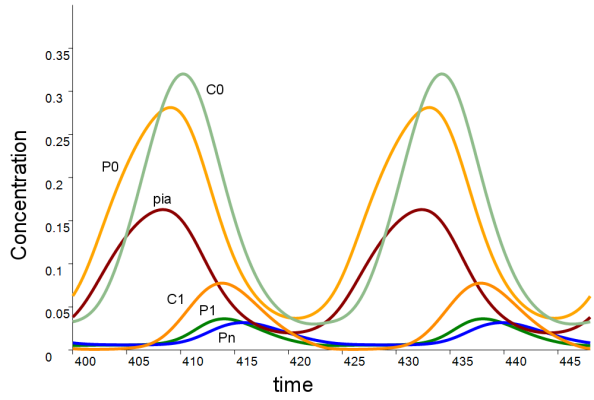
Name	Value	Name	Value	Name	Value
n	2	d_9	0.01113	V_δ	2.3363
k_8	0.9456	d_{10}	0.01113	$V_{\delta 2}$	1.113
k_{-8}	0.1113	b_0	0.002225	V_π	1.893
k_{10}	0.3338	b_4	0.01113	$V_{\pi 2}$	0.7788
k_{-10}	0.6675	b_7	0.005563	V_0	3.338
d_0	0.4033	s_1	0.89	V_4	8.344
d_1	0.4005	s_5	0.89	V_7	8.344
d_2	1.78	s_8	0.7788	k_5	2.0164
d_3	0.4033	J_δ	2	k_{-5}	3.025
d_4	1.00125	$J_{\delta 2}$	2	k_{in}	0.5041
d_5	0.8066	J_π	0.001	k_{out}	0.1008
d_7	0.7788	$J_{\pi 2}$	0.001	π_i	1
d_8	0.5006	K_i	0.001	δ	1

Table 4.3: M3 final parameter values. Final values reflect a scaling up in order to make a $\approx$ 24 hr. oscillation. Note that these are *slightly* different than the actual values, since they are rounded to 4 or 5 significant digits. This made no change to the rhythm other than a slight phase shift (of about 5-10 minutes).

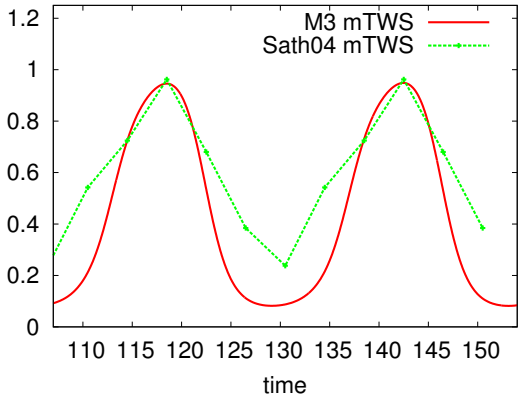
Our goal for this model is simply to replace some of the phosphatase activity caused by the addition of *tws* with the sequestering of PER into a complex with TIM. In the PER-TIM complex, PER is still available to be phosphorylated, but it is not degraded as a consequence. Therefore, the complexes C_0 and C_1 provide a safe haven for PER where it can acquire the phosphate group needed for inhibition without being eaten by the proteosome. This translates to assuming that the equilibrium for PER + TIM$\longrightarrow C_0$ lies toward C_0. As it turns out, this model requires no formal bifurcation optimization, just twiddling. We can slowly add in higher and higher values of V_7, s_8 and k_8 while lowering V_4 and s_4, ensuring that the binding and release into and out of the complexes provides the desired sequestering. We must also keep in mind that we do not want to totally replace the function of PP2A, since we would like to simultaneously satisfy our above goals as well.

4.3.1 Discussion of M3 Results

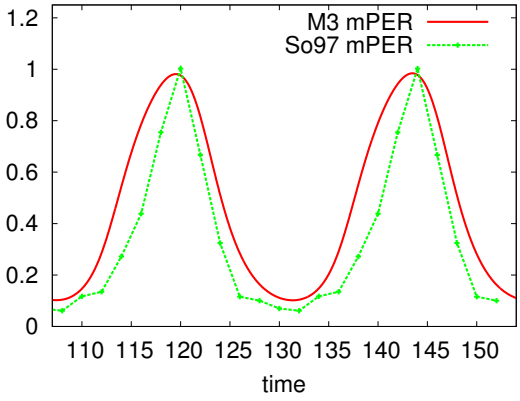
Acceptable results were obtained for this model with only hand twiddling. In Figure 4.9 we see decent approximation to details of the mPER, total PER, mTIM rhythms. Unfortunately,



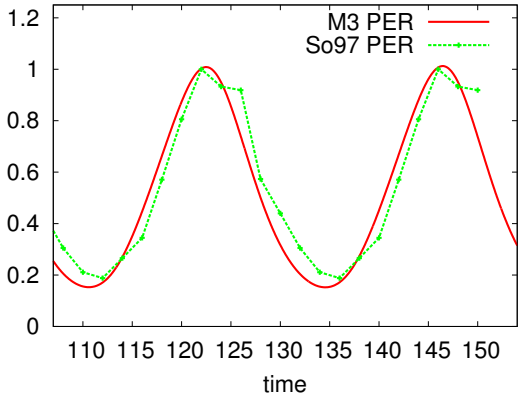
(a) Final TS



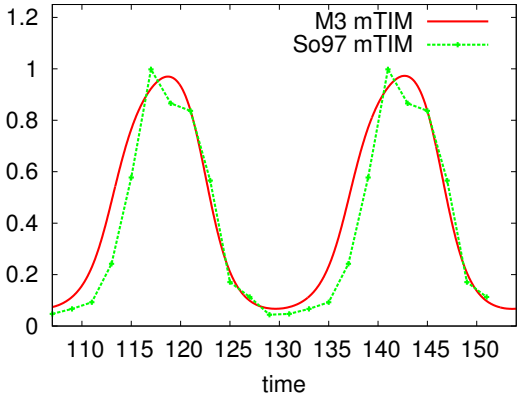
(b) mTWS: M3 vs. Experiment



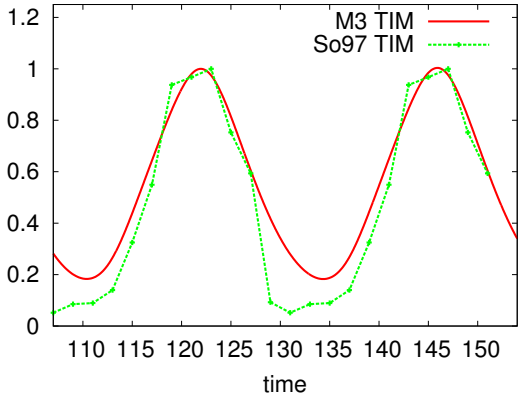
(c) mPER: M3 vs. Experiment



(d) PER: M3 vs. Experiment



(e) mTIM: M3 vs. Experiment



(f) TIM: M3 vs. Experiment

Figure 4.9: We see decent approximation to details of the mPER (c), total PER (d), mTIM (e) rhythms. Again, mTWS (b) has its minimum a bit low while TIM (f) has it a bit high in the simulations. These are details that can be remedied using quantitative fitting. Note that data is scaled to get roughly matching levels. In addition, the offset for the time-axis is slightly different between mPER and PER (our delay between mPER and PER is about 3 hours, instead of about 4 as in experiments). In (a), note that C_0 and C_1 are on par with the action of π_a, so phosphatase activity and sequestering of PER are both in part responsible for keeping enough PER around to get P_1 and finally P_n.

mTWS still has its minimum a bit low and it appears that TIM has its minimum a bit high in the simulations. However, these were considered minor details and again, with quantitative optimization, we would likely resolve such issues in short order.

We note, however, that the characteristic delay between mPER and PER maxima is smaller than it should be, coming in at about 3 instead of 4 hours (this is not apparent in the graphs, as we used slightly different time offsets in order to compare the quality of each rhythm separately). On the other hand, our mTIM$\longrightarrow$TIM delay comes out just about right at 4 hours (the delays are relatively clear in experimental data, see Figure 1.1(a)). Let us not dawdle any longer on M3; we have a much more fun model to try to get working in the next section.

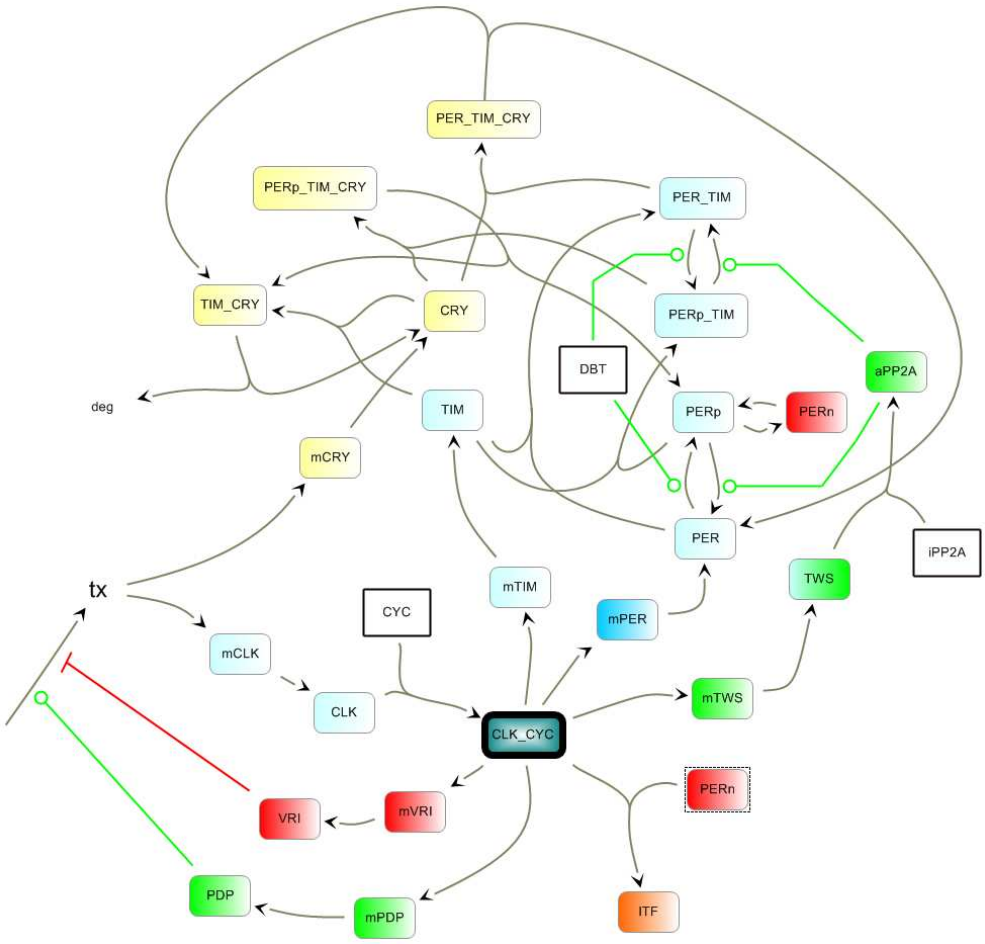


Figure 4.10: M4. This takes model M3 up a notch to include detail related to the *vri*, *pdp1ε*, *clk* and *cry* genes. *clk* dynamics completely replace the method of inhibition from the previous model in conjunction with *vri* and *pdp1ε* while *cry* gives us molecular detail concerning light input to the clock. Although this is a significant jump in complexity, i.e., from 11 to 24 state variables and from 39 to 75 parameters, the tools and methods described within this dissertation help us to fit the model. See equations (4.6) through (4.11) for the ODEs used to analyze this hypothesis.

4.4 M4

M4, our final model, is quite a bit more complex than M3 (see Figure 4.10). We have added the dynamics of *vri*, *pdp1ε*, *clk* and *cry*, all at once. Brave or stupid? With the help of the tools discussed in this dissertation, we can accomplish such increases in complexity (in an afternoon!). The first 11 equations for M4 are almost identical to M3, except for the addition of terms related to light (λ) in the equations for P_0, P_1, T_0, C_0 and C_1 and also the binding of P_n into the inactive transcription factor (I). The light terms added to the equations for P_0 and P_1 result from the release from CRY containing complexes as TIM is degraded, whereas the terms added to the equations for T_0, C_0 and C_1 result from TIM's

light dependent binding of CRY (C) whereby TIM is degraded.

$$\frac{dM_P}{dt} = b_0 + \frac{V_0 F^n}{J_f^n + F^n} - d_0 M_P \tag{4.6a}$$

$$\frac{dP_0}{dt} = s_1 M_P - \delta \frac{V_\delta P_0}{J_\delta + P_0} + \pi_a \frac{V_\pi P_1}{J_\pi + P_1} - k_8 P_0 T_0 + k_{-8} C_0 + \lambda C_3 - d_1 P_0 \tag{4.6b}$$

$$\frac{dP_1}{dt} = \delta \frac{V_\delta P_0}{J_\delta + P_0} - \pi_a \frac{V_\pi P_1}{J_\pi + P_1} - k_{10} P_1 T_0 + k_{-10} C_1 + \lambda C_4$$

$$-k_{in} P_1 + k_{out} P_n - d_2 P_1 \tag{4.6c}$$

$$\frac{dP_n}{dt} = k_{in} P_1 - k_{out} P_n - k_{13} P_n F + k_{-13} I - d_3 P_n \tag{4.6d}$$

$$\frac{dM_{T_\pi}}{dt} = b_4 + \frac{V_4 F^n}{J_f^n + F^n} - d_4 M_{T_\pi} \tag{4.7a}$$

$$\frac{dT_\pi}{dt} = s_5 M_{T_\pi} - k_5 \pi_i T_\pi + k_{-5} \pi_a - d_5 T_\pi \tag{4.7b}$$

$$\frac{d\pi_a}{dt} = k_5 \pi_i T_\pi - k_{-5} \pi_a \tag{4.7c}$$

$$\frac{dM_T}{dt} = b_7 + \frac{V_7 F^n}{J_f^n + F^n} - d_7 M_T \tag{4.8a}$$

$$\frac{dT_0}{dt} = s_8 M_T - k_8 P_0 T_0 + k_{-8} C_0 - k_{10} P_1 T_0 + k_{-10} C_1 - \lambda T_0 C - d_8 T_0 \tag{4.8b}$$

$$\frac{dC_0}{dt} = k_8 P_0 T_0 - k_{-8} C_0 - \delta \frac{V_{\delta 2} C_0}{J_{\delta 2} + C_0} + \pi_a \frac{V_{\pi 2} C_1}{J_{\pi 2} + C_1} - \lambda C_0 C - d_9 C_0 \tag{4.8c}$$

$$\frac{dC_1}{dt} = \delta \frac{V_{\delta 2} C_0}{J_{\delta 2} + C_0} - \pi_a \frac{V_{\pi 2} C_1}{J_{\pi 2} + C_1} + k_{10} P_1 T_0 - k_{-10} C_1 - \lambda C_1 C - d_{10} C_1 \tag{4.8d}$$

The next set of equations describes the kinetics of *clk* mRNA (M_K), protein (K) and the CLK-CYC transcription factor in active (F) and (I) forms. CYC, a constitutive component is represented in these equations by ξ. It is not readily clear how to simultaneously express both the inhibition by VRI (V) and the activation by PDP1ε (E) on the transcription of mCLK (we do not have a specific molecular hypothesis either about why *pdp1ε* products lag behind *vri* products). Although Cyran et al. (2003) find that VRI and PDP1ε compete to bind the *clk* promoter, we choose to use the simplest form involving independent terms for

each effect. Future models may choose a more sophisticated form for these kinetic effects.

$$\frac{dM_K}{dt} = b_{11} + \frac{V_{11e}E^n}{J_e^n + E^n} + \frac{V_{11v}}{1 + (V/J_v)^n} - d_{11}M_K \tag{4.9a}$$

$$\frac{dK}{dt} = s_{12}M_K - k_{12}\xi K + k_{-12}F - d_{12}K \tag{4.9b}$$

$$\frac{dF}{dt} = k_{12}\xi K - k_{-12}F - k_{13}P_nF + k_{-13}I - d_{13}F \tag{4.9c}$$

$$\frac{dI}{dt} = k_{13}P_nF - k_{-13}I - d_{14}I \tag{4.9d}$$

The next 6 equations describe the kinetics of *vri*, *pdp1ε* and *cry*. While *vri* and *pdp1ε* kinetics are simple and depend on the same activation mechanism as *per* and *tim*, *cry* contains the unsupported (to my knowledge) hypothesis that *cry* shares a promoter with *clk*. Experimental evidence, however, does show that *clk* and *cry* mRNA cycle in similar manner, out of phase with CLK-CYC controlled genes (such as *per* and *tim*). This is the type of hypothesis that a modeler might make when trying to imagine how unknown experimental details might be realized in the system. For the purposes of this research, it is irrelevant whether or not we are correct. Only time will tell whether such a hypothesis has any basis in fact.

$$\frac{dM_V}{dt} = b_{15} + \frac{V_{15}F^n}{J_f^n + F^n} - d_{15}M_V \tag{4.10a}$$

$$\frac{dV}{dt} = s_{16}M_V - d_{16}V \tag{4.10b}$$

$$\frac{dM_E}{dt} = b_{17} + \frac{V_{17}F^n}{J_f^n + F^n} - d_{17}M_E \tag{4.10c}$$

$$\frac{dE}{dt} = s_{18}M_E - d_{18}E \tag{4.10d}$$

$$\frac{dM_C}{dt} = b_{19} + \frac{V_{19e}E^n}{J_e^n + E^n} + \frac{V_{19v}}{1 + (V/J_v)^n} - d_{19}M_C \tag{4.10e}$$

$$\frac{dC}{dt} = s_{20}M_C + \lambda(C_2 - T_0C - k_{20}C) - d_{20}C \tag{4.10f}$$

Notice the $\lambda(C_2 - T_0C - k_{20}C)$ term in equation (4.10f). CRY binds TIM in a light dependent manner (into the TIM-CRY complex, C_2) and is quickly released again when TIM is degraded because CRY appears to have slower, light-dependent degradation kinetics. Busza et al. (2004) show that although a 10-min pulse of light commits TIM to degradation (and for this

pulse length, CRY and TIM have similar degradation kinetics), longer light pulses increase CRY degradation, but only in the continued presence of light. Hence, the $-\lambda k_{20} C$ term. Note that we are also assuming that the relative kinetics between the binding and release of CRY (with TIM and other complexes) are the same (in the following equations as well).

The last set of equations deals with the CRY complexes, which are only formed in the presence of light (hence, any formation terms will be multiplied by λ). TIM-CRY (C_2) is used in our model as the only complex from which TIM is degraded. PER-TIM-CRY (C_3) and PER-P-TIM-CRY (C_4) rapidly form in the presence of light and just as rapidly dissociate in to PER + TIM-CRY and PER-P + TIM-CRY, enabling the degradation of TIM via the mechanism described for TIM-CRY.

$$\frac{dC_2}{dt} = \lambda(T_0 C + C_3 + C_4 - C_2) - d_{21} C_2 \tag{4.11a}$$

$$\frac{dC_3}{dt} = \lambda(C_0 C - C_3) - d_{22} C_3 \tag{4.11b}$$

$$\frac{dC_4}{dt} = \lambda(C_1 C - C_4) - d_{23} C_4 \tag{4.11c}$$

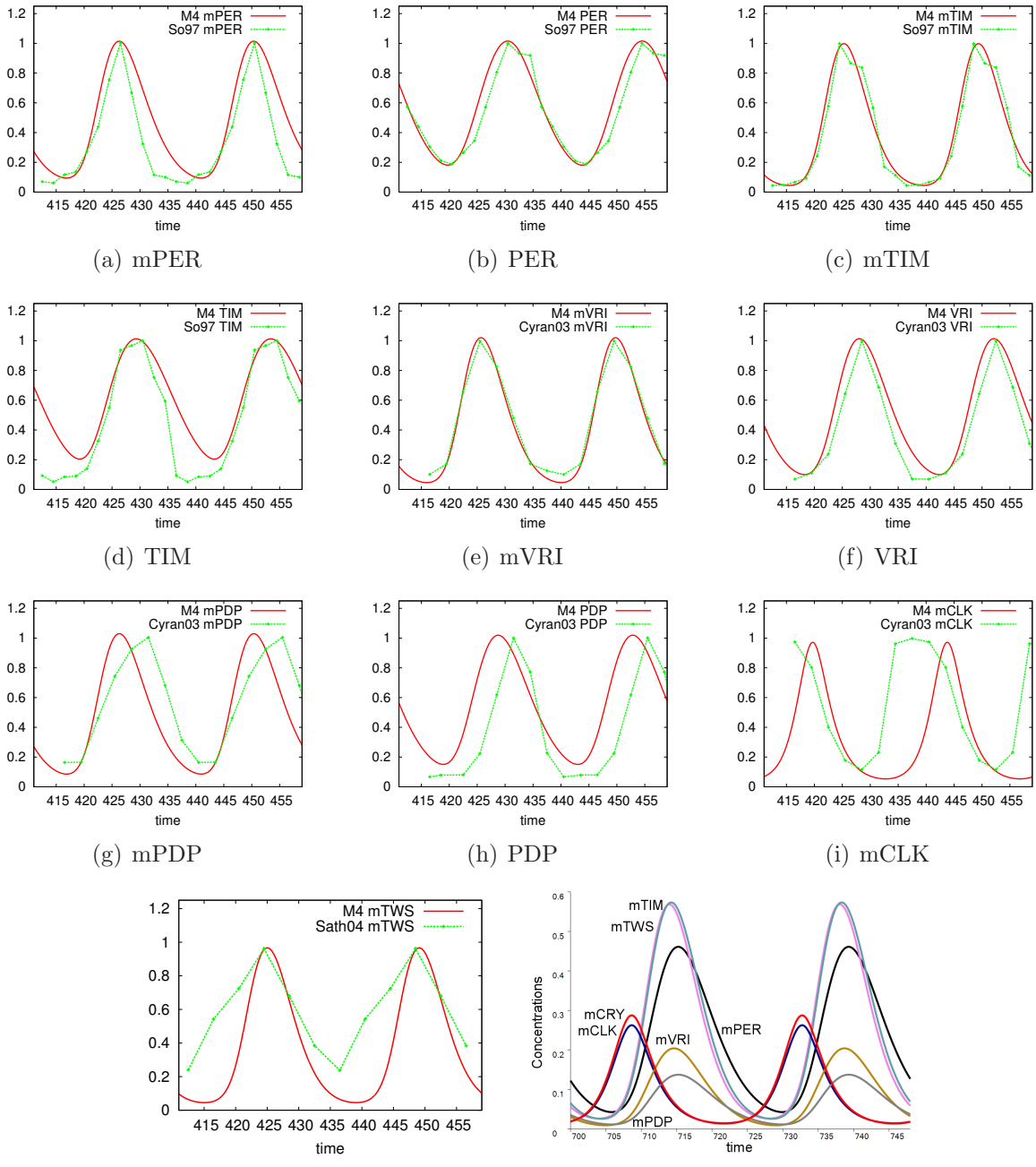
We note here that we have ignored dissociation of the CRY complexes into any species containing TIM. This is equivalent to assuming that the minimum pulse of light we will consider valid is 10 min (committing TIM to degradation). If shorter light pulses are to be considered, additional terms must be added allowing an escape for TIM.

4.4.1 Discussion of M4 Results

Given the degree of complexity we have just added to our model (we went from 11→24 state variables and from 39→75 parameters), we might just despair! On the other hand, we know that we had a working oscillator which will continue to operate as before assuming we can get the CLK-CYC dynamic to operate similarly to the previous inhibition term expressed directly with dependence on nuclear PER. The first attack is to choose reasonable (not necessarily easy) guesses for the unknown parameters and start modifying values so that, at the very least, we get steady state levels into reasonable ranges. Once this is done, we can do a random search for nearby parameter sets that give oscillation using the bifurcation

Name	Value	Name	Value	Name	Value
n	2	V_{15}	1.45	d_3	0.292379
λ	0	V_{17}	0.725	d_4	0.725906
s_1	0.64525	V_{19e}	0.4	d_5	0.584757
s_5	0.64525	V_{19v}	0.4	d_7	0.58
s_8	0.58	k_5	1.4619	d_8	0.58
s_{12}	0.3625	k_{-5}	2.19284	d_9	0.00806562
s_{16}	0.145	k_8	0.58	d_{10}	0.00806562
s_{18}	0.0725	k_{-8}	0.18125	d_{11}	0.3625
s_{20}	0.035	k_{10}	0.145	d_{12}	0.3625
J_δ	2	k_{-10}	0.435	d_{13}	0.3625
$J_{\delta 2}$	2	k_{12}	0.0725	d_{14}	0.0725
J_π	0.001	k_{-12}	0.00725	d_{15}	0.435
$J_{\pi 2}$	0.001	k_{13}	2.175	d_{16}	0.3625
J_f	0.1	k_{-13}	0.0725	d_{17}	0.29
J_v	0.005	k_{20}	0.01	d_{18}	0.3625
J_e	1	b_0	0.00161313	d_{19}	0.3625
V_δ	1.69378	b_4	0.00806562	d_{20}	0.005
$V_{\delta 2}$	0.806562	b_7	0.00403281	d_{21}	0.01
V_π	1.37116	b_{11}	0.000725	d_{22}	0.00725
$V_{\pi 2}$	0.564594	b_{15}	0	d_{23}	0.00725
V_0	2.41969	b_{17}	0	k_{in}	0.365474
V_4	6.04922	b_{19}	0	k_{out}	0.0730945
V_7	5.075	d_0	0.292379	π_i	1
V_{11e}	0.3625	d_1	0.290363	δ	1
V_{11v}	0.3625	d_2	1.2905	ξ	1

Table 4.4: M4 final parameter values. Once again, they are not neat given 1) they are partially derived from the previous (not-so-neat) set and 2) they are scaled to get a 24 hour rhythm.



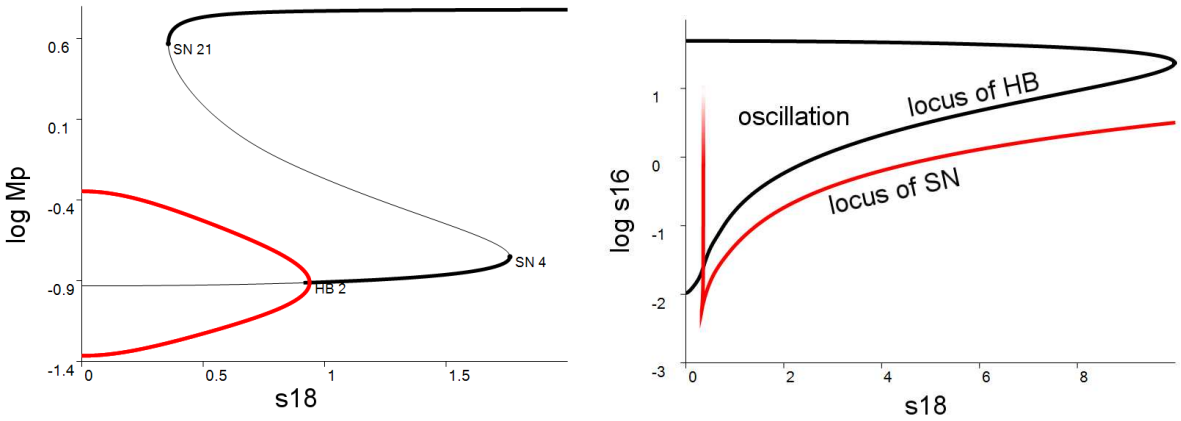
(a) mPER (b) PER (c) mTIM

(d) TIM (e) mVRI (f) VRI

(g) mPDP (h) PDP (i) mCLK

(j) mTWS (k) mRNA Comparison

Figure 4.11: We see decent approximation to details of the *per* (a)+(b), *tim* (c)+(d) and *vri* (e)+(f) mRNA and protein rhythms. There is *only* scaling the get the levels to match, so that we can compare time lags appropriately to properly assess the problems with this model. Since the peaks of *per* and *tim* mRNA and proteins are matched quite well, the respective delays are better than in M3. On the other hand, mPDP, PDP and mCLK are not correct, a result of us not properly fleshing out a hypothesis for the delay seen in mPDP relative to mVRI. (k) gives us the relative rhythms of the mRNAs, so we can see that mCLK and mCRY are not quite in anti-phase to the rest, but are at least halfway there.



(a) s_{18} (b) s_{16} vs. s_{18}

Figure 4.12: (a) s_{18} is the translation of mPDP to PDP, and therefore represents a positive feedback in the *clk* loop. This bifurcation diagram is similar to other positive feedback components in the system. (b) Two-parameter bifurcation diagram of $log(s_{16})$ vs. s_{18}. The region of oscillation outlined by the HB locus is unaffected by the interfering bistable region in large part (as can be seen in (a)), although it is possible to kick the system into steady state behavior by perturbing the state.

searching feature of Oscill8 (as described in Section 2.3.4). Indeed, this attack is fruitful and we can avoid any other optimization just by clever hand twiddling (the final parameter values are given in Table 4.4).

In Figure 4.11, we see the results of getting the model as well fit as can be expected given 1) no quantitative optimization to time series data and 2) a lack of a hypothesis concerning the difference in regulation of *vri* and *pdp1ε*. Subfigures (a) through (f) show that we have gotten *per*, *tim* and *vri* dynamics pretty well accounted for. mCLK, which depends heavily on the the dynamics of *pdp1ε*, is quite a bit off. If we could add a hypothesis about why *pdp1ε* dynamics are so delayed compared to *vri*, mCLK would likely come as a bonus given its dependence on *vri* and *pdp1ε*. All in all, the result are very satisfying.

An additional point of interest is the local dynamical structure. Although it is often a challenge to ensure large enough domains of oscillation, we find that the parameters listed in Table 4.4 put M4 in just such a place. Close by in a few parameters related to various positive feedback terms, we do jump out of the domain of oscillation as we move into bistable regions (see Figure 4.12(a)). This is exactly what we would like in a system that should be both robust to noise but sensitive to choice perturbation. Naturally, given the complexity of both M3 and M4, we could put together practically any dynamical structure we like nearby (and this was explored for both models), this region of parameter space gives appealing

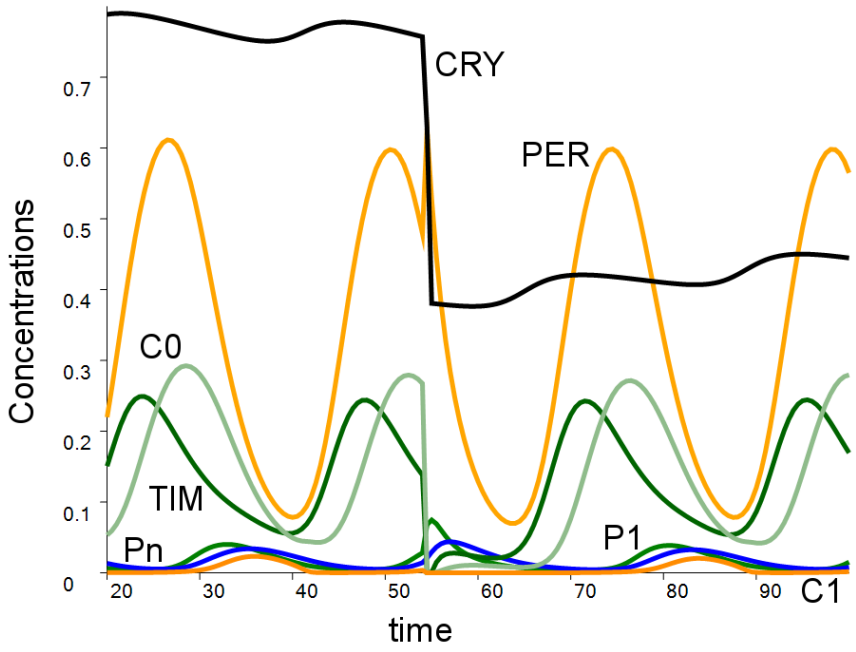


Figure 4.13: (a) At $t = 54$, lights go on ($\lambda \to 100$) until $t = 55$. As a result the rhythm is delayed by about an hour. Notice that CRY is in a high state, having built up in the absence of light. The immediate effects are obvious and logical: TIM is rapidly degraded, PER-TIM (C_0) and PER-P-TIM (C_1) are snatched up, TIM is degraded out and PER (P_0) and PER-P (P_1) are released.

dynamical structures that could easily lead to a discussion of mutant behaviors, etc.

Note on Light Effects

This model does pay some attention to the molecular mechanism of CRY activated TIM degradation, so we can run tests to see how pulses of light affect the rhythm. In Figure 4.13, we see that a light pulse of 1 hour given at $t = 54$ (for $\lambda = 100$) has the desired effects on the rhythm. Namely, TIM is swiftly degraded, and complexes including TIM (C_0 and C_1) rapidly dissociate and return the non-TIM constituents back to their respective pools (hence the abrupt increases in P_0 and P_1). Prior to lights on, CRY has built up to a constitutively high levels due to total darkness (this is experimentally accurate), but it is degraded fairly swiftly while lights are on. The net effect of the pulse of light is a slightly more than one hour phase advance in the rhythm. With a more solid hypothesis for *vri* vs. *pdp1ε* dynamics, a thorough study of the effects of light in this model would be quite interesting.

4.5 Summary

To conclude, enough detail is known about the circadian clock to make sufficiently complex molecular models. It turns out that building a model up piece by piece can be straight forward if you have the right tools and techniques. Honestly speaking, reading through the literature and deriving hypotheses about clock function is, and absolutely *should* be, the most time consuming part of the modeling process. For the models derived above, the hypotheses come from a large body of scientific literature (both computational and experimental) and a few novel ideas. The accumulation of such knowledge can take years of effort on the part of many researchers. On the other hand, the modeling work responsible for getting us from M0 to M4 took on the order of a week, once all the hypotheses, tools and techniques were in place. This is the greatest satisfaction that this research has not been done in vain! Although none of the models presented in this chapter are "complete" in the sense that they quantitatively reproduce all known experimental data, they are not far away. I am convinced that following up with more quantitative optimization and attention to detail would remedy these faults.

Chapter 5

Temperature Compensation

5.1 Overview

Temperature compensation is the curious property of the circadian clock that keeps the period (T) of the oscillator relatively constant over a large range of temperatures. Rate constants depend on temperature (θ) according to Arrhenius' law, $k_i = \alpha_i e^{-E_i/R\theta}$, where R is the universal gas constant, α_i determines the value of k_i at $\theta = 298$ K, and $E_i > 0$ is the activation energy of the i^{th} reaction. Some years ago, Ruoff pointed out (Ruoff et al., 1997) that a model of the circadian clock would be temperature compensated if

$$\frac{dT}{d\theta} = \sum_i \frac{\partial T}{\partial k_i}\frac{\partial k_i}{\partial \theta} = \frac{1}{R\theta^2}\sum_i \left(\frac{\partial T}{\partial k_i}\cdot \alpha_i E_i\right) \approx 0 \tag{5.1}$$

This sum can always be set close to zero by choosing the E_i's to properly weight the positive and negative terms of the sum (since $\partial T/\partial k_i > 0$ for some i and < 0 for others). However, the balance in eq. (5.1) is quite sensitive to changes of α_i and E_i, which represent the thermodynamic properties of individual reactions and would be the targets of random variation within a population. It is well known that the variation of circadian period within a species between individuals is very low, with a CV $= \frac{\text{std. dev. of period}}{\text{mean of period}}$ of about 4% (Rothenfluh et al., 2000a; Bao et al., 2001). Furthermore, it is also known that many mutants retain the ability to temperature compensate (Rothenfluh et al., 2000a; Morgan and Feldman, 2001) even though the basal period of the mutant may be different from 24 hr. These two facts

Model	**Compensation Relation**
LG	$v_m = 0.43 \cdot (k_s - 0.3)^{-0.16}$
TH	$J_p = .000032/(k_m^{3.2})$
RS	$\sigma = e^{-24.1 \cdot \mu}$

Table 5.1: Compensation relations used in tests CPV and CTD. A change to one of the independent parameters determines the value of the dependent parameter in order to keep the period of oscillation fixed.

lead us to consider the alternate possibility that evolution has optimized the mechanism such that $\partial T/k_i \approx 0$ for most i instead of exquisitely fine-tuning the α_i and E_i. In this case, control of the period is concentrated into a few choice constants and the period becomes very robust to generic mutation (though more sensitive to specific mutation, fitting the current sentiment about robustness vs. sensitivity (Carlson and Doyle, 1999, 2002)).

As it turns out, most published models of the circadian clock rely on a negative feedback mechanism to produce limit cycle oscillations via a Hopf bifurcation (see Section 1.3), and the period of limit cycle oscillators tend to depend rather sensitively on many of the kinetic constants k_i (see Figure 5.3). In Hong et al. (2006), we describe a mechanism, called the resetting hypothesis, that lends itself to robust temperature compensation. We compare the robustness of circadian period of the resetting model, RS (Hong et al., 2006), to two published limit cycle models, LG (Leloup and Goldbeter, 1998) and TH (Tyson et al., 1999)[1], using the following two tests:

1. **Test I** is designed to measure variability of CR period with respect to simultaneous random perturbations of all the kinetic constants in the model (i.e., variability across individuals). A robustly compensated model must not lose its ability to compensate for small random variations. We give two flavors of this test, one where we randomly perturb all parameters to assess the inherent sensitivity of the model to perturbation and one where, in addition to the random perturbations, we allow a compensation relation between the two most period-sensitive parameters. See Table 5.1 for the compensation relations used.

[1]See appendix A for ODE files for each of these models and Tables 5.2, 5.3 and 5.4 for parameter values and their associated enthalpy and energies.

p	$p(298)$	ε	s
v_s	0.76	9	-9
v_m	0.65	10	-10
K_i	1	19	19
K_m	0.5	13	12.3069
n	4	0	0
k_s	0.38	11	-11
V_1	3.2	11	-11
K_1	2	8	8.69315
V_2	1.58	13	-13
K_2	2	12	12.6931
V_3	5	7	-7
K_3	2	10	10.6931
V_4	2.5	10	-10
K_4	2	6	6.69315
k_{in}	1.9	10	-10
k_{out}	1.3	6	-6
v_d	0.95	10	-10
K_d	0.2	7	5.39056

Table 5.2: Basal parameter values for LG, including reasonable estimates for ε and s.

2. **Test II** is designed to measure a model's ability to maintain temperature compensation in the face of single mutations (which cause random changes in α_i and E_i for some i). A robustly compensated model should handle mutations to many parameters perfectly (with possible aberrant basal periods), while some parameters will break compensation badly.

For each test, we generate a large sample of randomly perturbed individuals. In Test I, an "individual" is generated by multiplying each basal parameter value (though not violating the compensation relation) by a new random number drawn from $\mathcal{N}(1, \sigma_p)$, the normal distribution with mean 1.0 and standard deviation σ_p. In Test II, a "mutant" organism is created by randomly selecting a rate constant $k_i = \alpha_i e^{-E_i/R}$ and altering both α_i and E_i by random multiplicative factors drawn from $\mathcal{N}(1, \sigma_p)$. Then the mutant organism's period is computed for $\theta = 293, 294, \ldots, 303$ K and its ability to temperature compensate is measured as $\Delta T = T_{max} - T_{min}$. Both tests are run for many values of σ_p between 0.01 and 0.4 and the results plot the coefficient of variation of the period, CV $= \frac{\text{std. dev. of period}}{\text{mean of period}}$ (for Test

p	$p(298)$	ε	s
v_m	1	4	-4
k_m	0.1	3	-3
v_p	0.5	11	-11
k_{p1}	10	6	-6
k_{p2}	0.03	13	-13
k_{p3}	0.1	4	-4
J_p	0.05	5	2.00427
P_{crit}	0.1	19	16.6974
K_{eq}	200	7	12.2983

Table 5.3: Basal parameter values for TH, including reasonable estimates for ε and s.

p	$p(298)$	ε	s
v_m	1	4	-4
k_m	0.1	3	-3
v_p	0.5	11	-11
k_{p1}	10	6	-6
k_{p2}	0.03	13	-13
k_{p3}	0.1	4	-4
J_p	0.05	5	2.00427
P_{crit}	0.1	19	18.4892
K_{eq}	200	7	7
μ	0.0288	10	-10
σ	0.5	-6.9315	-7.6246
$thresh$	2	0	0

Table 5.4: Basal parameter values for RS, including reasonable estimates for ε and s.

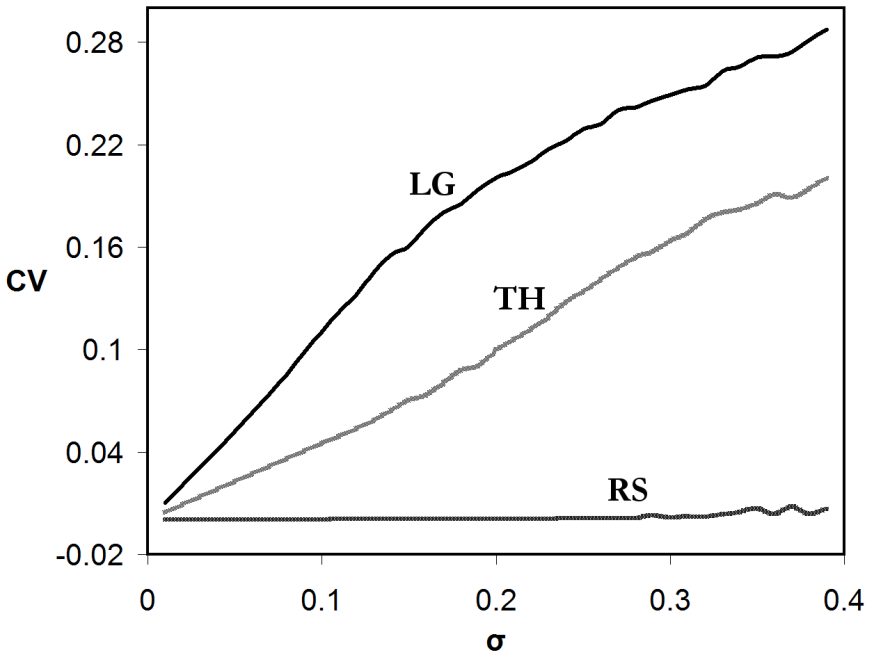


Figure 5.1: Results for Test I for each of LG, TH and RS. For each $\sigma_p = 0.01, ..., 0.4$, we calculate 10,000 perturbed individuals and calculate the coefficient of variation of the period with respect to σ_p, plotted here.

I, Figure 5.1) or the average value of ΔT versus σ_p (for Test II, Figure 5.2). Note that for Test II, the harmonic mean, $\langle \Delta T \rangle = n \left(\sum_{i=1}^{n} \Delta T_i^{-1} \right)^{-1}$, is used to measure the "average value" of ΔT because it is not dominated, as is the arithmetic mean, by a few large values of ΔT_i. Our rationale for doing so is that evolutionary selection pressure is likely able to deal with rare and extreme disturbances of temperature compensation (a few large values of ΔT_i), by eliminating the affected mutants, but is less able to deal with frequent upsets of the temperature compensation mechanism (many medium-sized values of ΔT_i). We will now describe the details of the algorithm used to calculate these results.

5.2 Algorithms for Tests I & II

First note that parameters come in basically two groups: a) rate constants and b) equilibrium constants. (Other parameters, such as Hill coefficients, are assumed to be independent of temperature for the sake of this discussion). These two groups have a different form for their

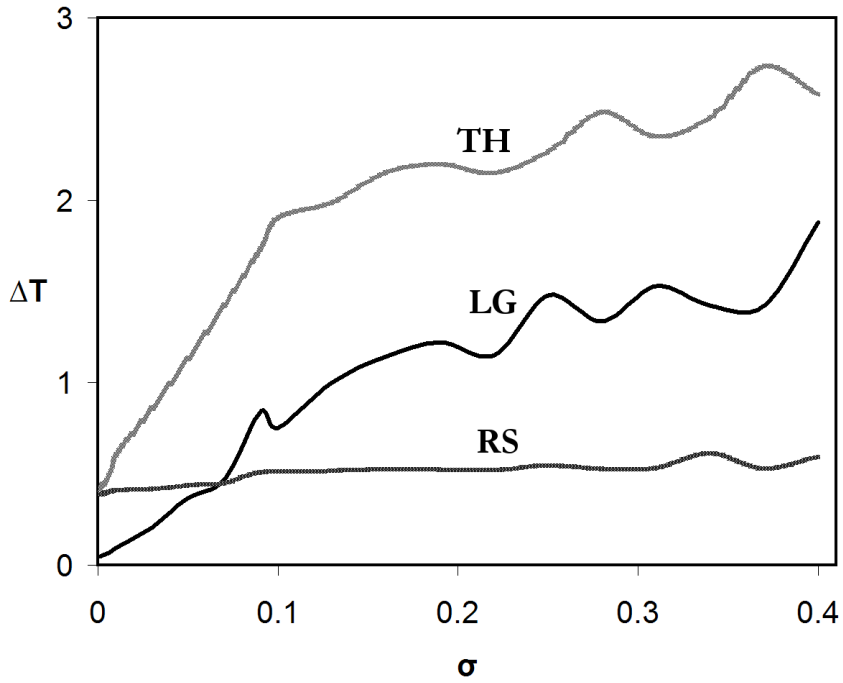


Figure 5.2: Results of Test II for each of LG, TH and RS. Both LG and TH show quite a large dependence on mutations, whereas RS is largely insensitive to such perturbations.

temperature dependence according to the law of Arrhenius' (for rate constants),

$$k_i = k_i^\circ e^{s_i^\dagger} e^{-\varepsilon_i^\dagger \cdot 298/\theta}, \tag{5.2}$$

and the Gibbs equation (for equilibrium constants),

$$K_i = e^{s_i^\circ} e^{-\varepsilon_i^\circ \cdot 298/\theta}, \tag{5.3}$$

both of which depend on the the two thermodynamic parameters, ε (energy) and s (enthalpy).

> Recall that ΔG is the free energy and satisfies $\Delta G = \Delta H - \theta \Delta S$, where ΔH is the enthalpy (basic heat content), ΔS is the entropy, and θ is the temperature. Also, because $\Delta H = \Delta E + P\Delta V$ (activation energy plus the change in volume times the pressure) and $\Delta V \approx 0$ in biochemical reactions, we can assume that $\Delta H \approx \Delta E$. So, setting $s = \Delta S/R$ and $\varepsilon = \Delta H/(298 \cdot R)$ gives us the relationships (5.2) and (5.3) from the usual forms for Arrhenius' law, $k_i^\circ = e^{-\Delta G_i^\dagger/R\theta}$, and Gibbs equation, $K_i = e^{-\Delta G_i^\circ/R\theta}$. Note also that for rate constants, $\varepsilon_i > 0$ and $s_i < 0$.

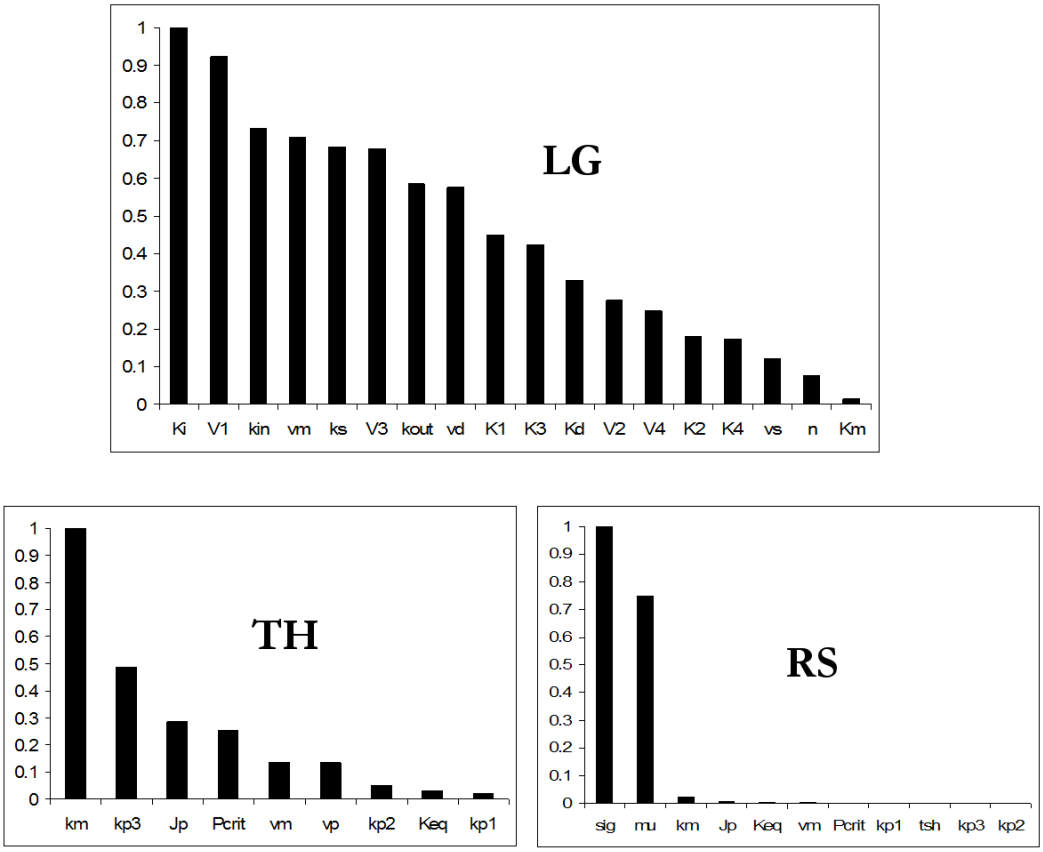


Figure 5.3: Sensitivity rankings for each of LG, TH and RS. Notice that LG has a much more even spread across its many parameters, whereas TH concentrates the period-sensitivity into a smaller group. RS concentrates all of its period-sensitivity into the two control parameters, by design.

For a fair comparison of the robustness of the period of the three models LG, TH and RS, we first need to equip LG and TH with compensation relations analogous to $\sigma = e^{-24\mu}$ (a constraint on the RS model). To this end, we first rank for each model the sensitivity coefficients, $\frac{\partial \log T}{\partial \log p_i}$, in order to identify the parameters that most influence the period (see Figure 5.3). It is worthwhile to note from Figure 5.3 that, while RS concentrates virtually all of the sensitivity into just two parameters (by design), both LG and TH have a more even distribution. In addition, the fact that LG has a more even distribution than TH is significant in the results (see Section 5.3). From these rankings, we choose as compensatory parameters: v_m and k_s for LG and J_p and k_m for TH. We then continue the limit cycle period in the two chosen parameters, generating a locus of (p_i, p_j) pairs that maintain $T =$ constant ≈ 24 hrs. From this parametric curve, we derive a compensation relation for p_i as

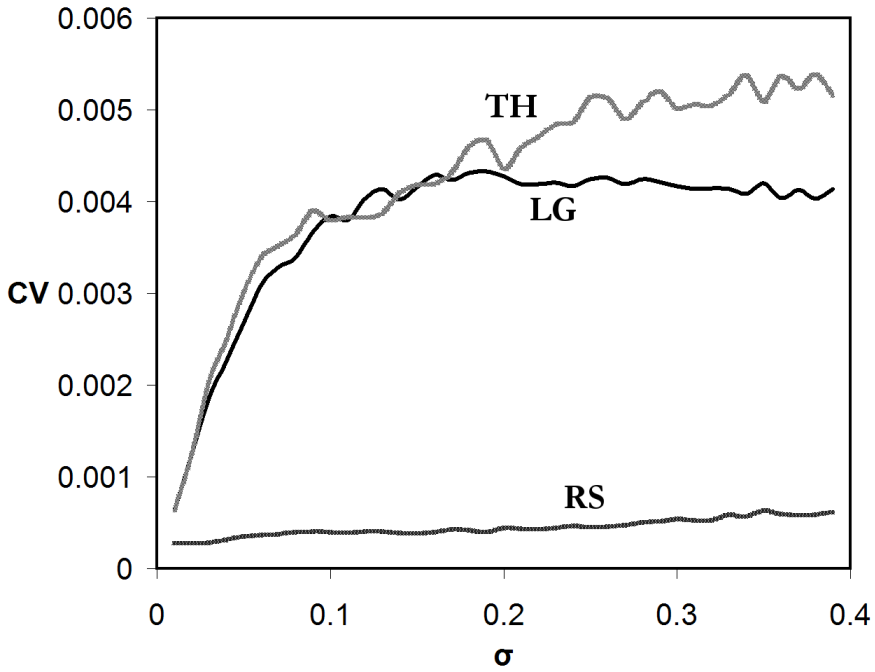


Figure 5.4: CV vs. σ_p over a 1000 random parameter samples for which we have perturbed only the independent parameter (and then compensated the dependent parameter) according to the relationships given in Table 5.1. Note that all models compensate to within less than about 1/2% of the basal period.

a function of p_j. For LG, we find that

$$v_m = 0.43 \cdot (k_s - 0.3)^{0.16}$$

gives constant period close to 24 hrs, and for TH,

$$J_p = 3.2 \times 10^{-5} \cdot k_m^{-3.2}.$$

These functions fix the period in two parameters to within about a half-a-percent (see Figure 5.4), so we may confidently begin our analysis with the three models on equal footing.

Test I aims to assess how the period of a model co-varies (measured by the coefficient of variation of the distribution of resulting periods, CV) in response to varying strengths, σ_p, of vector perturbations applied to the basal parameter set, simulating variations among individuals within a population. The algorithm by which this is accomplished is given in Figure 5.5. Because certain perturbations move us outside of the domain of oscillation, especially for higher perturbation strengths, we end up throwing out many samples in order

```
// initialize basal parameter vector of length m
b =< b_j >

// for many different perturbation strengths
for each σ_p = 0.01, ..., 0.4

  // collect 10000 individuals and their periods
  for i = 1 to 10000
    1) draw m random numbers from N(1, σ_p) to get r =< r_j >
    2) set the individual's parameters, p =< r_j · b_j >
    3) apply the compensation relation
    4) compute period and save
  end

  // calculate statistic; (CV, σ_p) is what we plot
  compute CV from the mean and variance of the collected periods
end
```

Figure 5.5: Algorithm, Test I. We collect a large number of individuals that are randomly perturbed (with strengths $\sigma_p = 0.01, \ldots, 0.4$) from some ideal wild-type parameter set and compute the resulting coefficient of variation of the parameter with respect to σ_p.

to get 10^4 valid, oscillating individuals. The number of samples thrown out indicates how unlikely the oscillation is to be preserved under given perturbation strength, providing another statistic of robustness that is unrelated to temperature compensation (see Figure 5.6).

Test II is meant to simulate a sample of single genetic mutations and measure the ability of the affected individuals to temperature compensate. Each rate constant's dependence on temperature is given by equation (5.2). Since values for $s_i^\dagger$ and $\varepsilon_i^\dagger$ are not known experimentally, we suppose (not unreasonably) that

$$s_i^\dagger = -\varepsilon_i^\dagger \tag{5.4}$$

and choose $\varepsilon_i^\dagger \in [3, 20]$ in order to obtain good temperature compensation for each model (see Tables 5.2, 5.3 and 5.4 for the values chosen and Section 5.3 for a brief discussion of how they were chosen). Because $s_i^\dagger$ and $\varepsilon_i^\dagger$ are dependent on the biophysical properties of

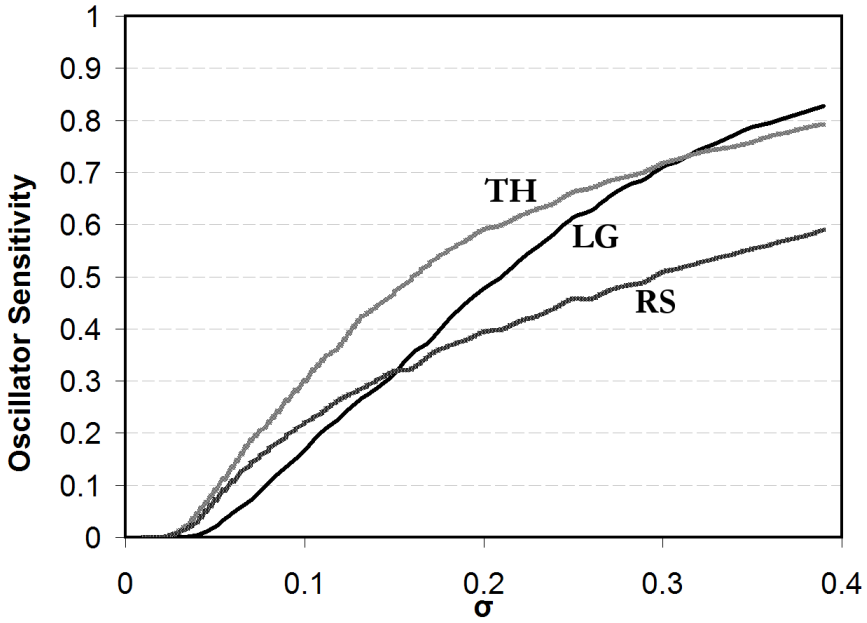


Figure 5.6: Plots the probability of an "individual" being perturbed outside of the oscillatory domain during Test I. This is a measure of how robust the domain of oscillation is for each model. Note that initially (for low σ_p), TH and RS (related models) loose oscillation more frequently than LG, but that for higher perturbation strengths, both TH and RS are more robust than LG.

the enzyme that catalyzes reaction i, their values are genetically determined and subject to random variations by single gene mutations.

Binding constants are treated similarly, where s_i° and ε_i° are the dimensionless standard entropies and enthalpies of the binding reaction. Note, however, that in this case,

$$s_i^\circ = \varepsilon_i^\circ + \ln K_i(298), \tag{5.5}$$

so there is no need to make an additional assumption such as that given in equation (5.4). These values are also subject to modification by mutations that change the binding properties of specific proteins.

With this in mind, we generate "mutant" individuals by perturbing s_i and ε_i for a particular choice of i. (Because the models have already been temperature compensated by our choices of ε_i, we do not constrain the parameters any further by the compensation relations of Test I). For each mutant individual, we calculate the maximum excursion of the period, $\Delta T = T_{max} - T_{min}$, over a 10° K temperature range ($293 < \theta < 303$) as a measure of the mutant's ability to temperature compensate. Although the variation of the period over

```
// initialize
a =< ε_i > // the basal vector of ε_i values, of length m
b =< s_i > // the basal vector of s_i values, of length m

// loop through a number of perturbation strengths, σ_p
for each σ_p = 0.01, ..., 0.4

  // for each parameter (that can be mutated)
  for i = 1 to m

    // collect 10 random samples of individuals with p_i-mutations
    // and their periods in the temperature range 293-303 K
    for j = 1 to 10
      1) draw r_1 and r_2 from N(1, σ_p)
      2) set the individual's ε = a, s = b
      3) perturb ε_i = a_i · r_1, s_i = b_i · r_2
      4) compute periods, T_k, for θ = 293, 294, ..., 303 K
      5) compute temp. comp. stat, ΔT_{i,j} = max{T_k} − min{T_k},
    end

    // calculate intermediate statistic
    compute ΔT_i = arithmetic mean of ΔT_{i,j} from p_i-mutation stat
  end

  // calculate final statistic; (ΔT, σ_p) is what we plot
  compute ΔT as the harmonic mean of ΔT_i
end
```

Figure 5.7: Algorithm, Test II.

the stated temperature range may not be strictly increasing or decreasing and may in fact be nonlinear, this measure was considered adequate given a sampling of perturbations for the range of σ_p under consideration. A more accurate statistic would depend on the shape of the curve in addition to the maximum and minimum values. The algorithm used to generate results for this test can be found in Figure 5.7.

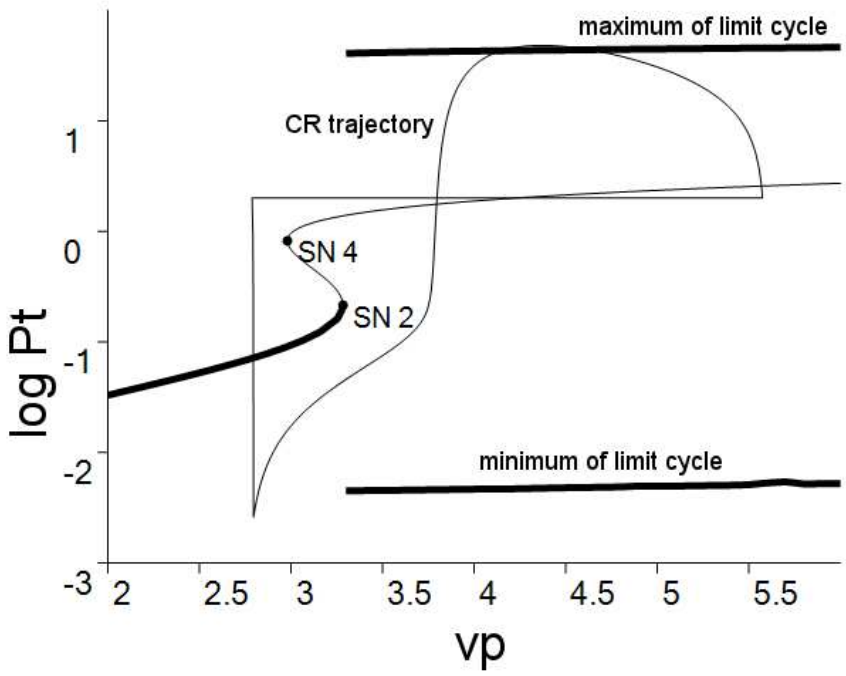


Figure 5.8: A one parameter bifurcation diagram depicting how the parameter v_p influences the dynamics of the RS model. Plotted on top of the v_p bifurcation curve is the natural, wild-type rhythm of the model, when v_p is treated as a dynamic variable.

5.3 Discussion

LG and TH are limit cycle oscillators, whereas the oscillatory behavior of RS relies on a resetting mechanism which switches the system between a limit cycle and a stable steady state. An important distinction between the limit cycle models and the resetting model is how the period of the oscillator depends on its many parameters. Whereas the limit cycle oscillators may depend quite sensitively on many of their parameters, the resetting model is constructed precisely so that there are only *two* sensitive parameters. Choosing just the right balance between these two parameters (see Table 5.1) guarantees a particular period, assuming other parameters do not stray too far from their basal values.

To understand the resetting model better, consider Figure 5.8. Here we can see that depending on the value of v_p (treating v_p as a parameter, rather than a state variable), there is a region of with a single stable steady state (low v_p) ending at a SNIC bifurcation (labeled as SN 2), after which there is a region of limit cycle oscillation (for high v_p). As long as changes in other system parameters keep the basic structure of this bifurcation diagram intact, choosing μ and σ so that $\sigma = e^{-\mu T}$ will guarantee a period of T. We can see this by

noting that $\dot{v}_p = \mu \cdot v_p$ (see Appendix A for a definition of the model and this differential equation) has the solution $v_p(t) = v_p(0)e^{\mu t}$, so that when we reset v_p by the factor $1/\sigma$ at time T, the result is $\frac{v_p(T)}{v_p(0)} = 1/\sigma$ or $\sigma = e^{-\mu T}$. To see that this is not the case for limit cycle oscillators, all we really need to do is consider the period-sensitivity ranking of the parameters for our three models given in Figure 5.3. The obvious things to notice here are that

- the RS model has virtually no sensitivity except for that attributed to μ and σ, the two parameters engineered to control the period. This predicts that RS should pass both Test I & II with flying colors (which it does, see Figures 5.1 and 5.2).
- both LG and TH have many more than two parameters that heavily influence the period (predicting the relatively poor results for them in Test I & II).
- LG has a much more even distribution of sensitivity compared to TH, so that we could have predicted that LG would fair worse than TH in Test I.

A Disclaimer for Activation Energy Estimates

Above, we mentioned that we somehow chose reasonable estimates for ε and s for each parameter such that the model is well temperature compensated over a 10° K range of values. Since experimental estimates for these values do not exist, we first employed a simple optimization algorithm to find a decent set, using $\Delta T = T_{max} - T_{min}$ as the objective function for a given sample. However, to keep things simple for publication, we disallowed non-integer values for ε (from which we derived s values according to equations (5.4) and (5.5)). Given this additional, discontinuous constraint, it turned out to be just as efficient to check about 1000 random samples and pick the one with the lowest ΔT. For LG, a set was found with $\Delta T = 0.046$ and for TH, a set was found with $\Delta T = 0.38$ (these data points can be seen in the limit $\sigma_p \to 0$ in Figure 5.2). Both of these were deemed good enough for purposes of this analysis. Admittedly, this process only determines a feasible set that gives temperature compensation and may have no relationship with actual experimental data, known or unknown. Nevertheless, they do provide a basis from which to begin our studies.

Software

Note that all of the software developed to perform these tests is included in the command line version of OSCILL8, using the "-tc" options.

Appendix A

ODE Files

A.1 Bifurcation Matching

A.1.1 Simple

```
# simple, non-biological model taken from Borisuk & Tyson, 1998

x'=p*(x*(1-x)*(1+x)-y)
y'=(x-a)*(b-y)-c

param p=1,a=-0.5,b=0.5,c=0.1

init x=1,y=1
```

A.2 Circadian Models

Here are the ODE files used for the circadian models presented in Chapter 4.

A.2.1 M0

```
# M0 from Emery Conrad's PhD Dissertation
# March 2006

#o8> possvs

dMp/dt = b0+V0/(1+(Pn/Ki)^n) - d0*Mp
dP0/dt = s0*Mp - Vd*DBT*P0/(Jd+P0) + Vp*PP2A*P1/(Jp+P1) - d1*P0
dP1/dt = Vd*DBT*P0/(Jd+P0) - Vp*PP2A*P1/(Jp+P1) - d2*P1 - kin*P1 + kout*Pn
dPn/dt = kin*P1 - kout*Pn - d3*Pn

init Mp = 0.1, P0 = 1, P1 = 0, Pn = 0

param n = 2
param DBT = 1
param PP2A = 1

param Jd = 1, Jp = 1, Ki = 1
param d0 = 0.1, d1 = 0.1, d2 = 0.1, d3 = 0.1
param kin = 1, kout = 1
param b0 = 0.1, s0 = 0.1
param Vd = 1, Vp = 1, V0 = 1
```

A.2.2 M1

```
# M1 from Emery Conrad's PhD Dissertation
# March 2006

#o8> possvs

dMp/dt = b0 + V0/(1+(Pn/Ki)^n) - d0*Mp
dP0/dt = s1*Mp - Vd*DBT*P0/(Jd+P0) + Vp*pia*P1/(Jp+P1) - d1*P0
dP1/dt = Vd*DBT*P0/(Jd+P0) - Vp*pia*P1/(Jp+P1) - d2*P1 - kin*P1 + kout*Pn
dPn/dt = kin*P1 - kout*Pn - d3*Pn

dMTpi/dt = b4 + V4/(1+(Pn/Ki)^n) - d4*MTpi
dTpi/dt  = s5*MTpi - k5*Tpi*pii + km5*pia - d5*Tpi
dpia/dt  = k5*Tpi*pii - km5*pia

init Mp = 0.1, P0 = 1, P1 = 0, Pn = 0
init MTPi = 0.1, Tpi = 1, pia = 0

param n = 2
param Jd = 0.2, Jp = .01, Ki = 0.01
param d0 = 0.2, d1 = 0.4, d2 = 0.1, d3 = 0.2, d4 = 0.1, d5 = 0.1
param kin = 0.05, kout = 0.05
param b0 = 0, s1 = 0.4, b4 = 0.1, s5 = 0.1
param Vd = 1, Vp = 0.5, V0 = 5, V4 = 1
param pii = 1, k5 = 0.1, km5 = 0.1
param DBT = 1
```

A.2.3 M3

```
# M3 from Emery Conrad's PhD Dissertation
# March 2006

#o8> possvs

dMp/dt = b0 + V0/(1+(Pn/Ki)^n) - d0*Mp
dP0/dt = s1*Mp - Vd*DBT*P0/(Jd+P0) + Vp*pia*P1/(Jp+P1) - k8*P0*T0 + km8*C0 - d1*P0
dP1/dt = Vd*DBT*P0/(Jd+P0) - Vp*pia*P1/(Jp+P1) - k10*P1*T0 + km10*C1 - kin*P1 + kout*Pn - d2*P1
dPn/dt = kin*P1 - kout*Pn - d3*Pn

dMTpi/dt = b4 + V4/(1+(Pn/Ki)^n) - d4*MTpi
dTpi/dt  = s5*MTpi - k5*Tpi*pii + km5*pia - d5*Tpi
dpia/dt  = k5*Tpi*pii - km5*pia

dMt/dt   = b7 + V7/(1+(Pn/Ki)^n) - d7*Mt
dT0/dt   = s8*Mt - k8*P0*T0 + km8*C0 - k10*P1*T0 + km10*C1 - d8*T0
dC0/dt   = k8*P0*T0 - km8*C0 - Vd2*DBT*C0/(Jd2+C0) + Vp2*pia*C1/(Jp2+C1) - d9*C0
dC1/dt   = Vd2*DBT*C0/(Jd2+C0) - Vp2*pia*C1/(Jp2+C1) + k10*P1*T0 - km10*C1 - d10*C1

init Mp = 0.1, P0 = 1, P1 = 0, Pn = 0
init MTPi = 0.1, Tpi = 1, pia = 0
init Mt = 0.1, T0 = 0, C0 = 0, C1 = 0

param n = 2
param k8 = 0.1, km8 = 0.1, k10 = 0.1, km10 = 0.1
param d0 = 0.2, d1 = 0.4, d2 = 0.1, d3 = 0.2, d4 = 0.1, d5 = 0.1
param d7 = 0.1, d8 = 0.1, d9 = 0.01, d10 = 0.01
param b0 = 0, b4 = 0.1, b7 = 0.1, s1 = 0.4, s5 = 0.4, s8 = 0.4
param Jd = 0.2, Jd2 = 0.2, Jp = .01, Jp2 = 0.1, Ki = 0.01
param Vd = 1, Vd2 = 1, Vp = 0.1, Vp2 = 0.1, V0 = 5, V4 = 1, V7 = 1
param k5 = 0.1, km5 = 0.1
param kin = 0.05, kout = 0.05
param pii = 1, DBT = 1
```

A.2.4 M4

```
# M4 from Emery Conrad's PhD Dissertation
# March 2006
# (Part 1 of 2)
#o8> possvs

dMp/dt = b0 + V0*F^n/(Jf^n+F^n) - d0*Mp
dP0/dt = s1*Mp - Vd*DBT*P0/(Jd+P0) + Vp*pia*P1/(Jp+P1) - k8*P0*T0 + km8*C0 + light*C3 - d1*P0
dP1/dt = Vd*DBT*P0/(Jd+P0) - Vp*pia*P1/(Jp+P1) - k10*P1*T0 + km10*C1 \
         + light*C4 - kin*P1 + kout*Pn - d2*P1
dPn/dt = kin*P1 - kout*Pn - k13*Pn*F + km13*I - d3*Pn

dMTpi/dt = b4 + V4*F^n/(Jf^n+F^n) - d4*MTpi
dTpi/dt  = s5*MTpi - k5*Tpi*pii + km5*pia - d5*Tpi
dpia/dt  = k5*Tpi*pii - km5*pia

dMt/dt = b7 + V7*F^n/(Jf^n+F^n) - d7*Mt
dT0/dt = s8*Mt - k8*P0*T0 + km8*C0 - k10*P1*T0 + km10*C1 - light*T0*C - d8*T0
dC0/dt = k8*P0*T0 - km8*C0 - Vd2*DBT*C0/(Jd2+C0) + Vp2*pia*C1/(Jp2+C1) - light*C0*C - d9*C0
dC1/dt = Vd2*DBT*C0/(Jd2+C0) - Vp2*pia*C1/(Jp2+C1) + k10*P1*T0 - km10*C1 - light*C1*C - d10*C1

dMk/dt = b11 + V11e*E^n/(Je^n+E^n) + V11v/(1+(V/Jv)^n) - d11*Mk
dK/dt  = s12*Mk - k12*CYC*K + km12*F - d12*K
dF/dt  = k12*CYC*K - km12*F - k13*Pn*F + km13*I - d13*F
dI/dt  = k13*Pn*F - km13*I - d14*I

dMv/dt = b15 + V15*F^n/(Jf^n+F^n) - d15*Mv
dV/dt  = s16*Mv - d16*V

dMe/dt = b17 + V17*F^n/(Jf^n+F^n) - d17*Me
dE/dt  = s18*Me - d18*E

dMc/dt = b19 + V19e*E^n/(Je^n+E^n) + V19v/(1+(V/Jv)^n) - d19*Mc
dC/dt  = s20*Mc + light*(C2 - T0*C - k20*C) - d20*C

dC2/dt = light*(T0*C+C3+C4 - C2) - d21*C2
dC3/dt = light*(C0*C - C3) - d22*C3
dC4/dt = light*(C1*C - C4) - d23*C4
```

```
# M4 from Emery Conrad's PhD Dissertation
# March 2006
# (Part 2 of 2)

init Mp = 0.1, P0 = 1, P1 = 0, Pn = 0
init MTPi = 0.1, Tpi = 1, pia = 0
init Mt = 0.1, T0 = 0, C0 = 0, C1 = 0
init Mk = .1, K = 1, F = 1, I = 0
init Mv = 0.1, V = 0
init Me = 0.1, E = 0
init Mc = 0.1, C = 0
init C2 = 0, C3 = 0, C4 = 0

param n = 2
param light = 0.001

param s1 = 0.4, s5 = 0.4, s8 = 0.4, s12 = 0.5,
param s16 = 0.5, s18 = 0.5, s20 = 0.5

param Jd = 0.2, Jd2 = 0.2, Jp = .01, Jp2 = 0.1
param Jf = 0.01, Jv = 0.01, Je = 0.01

param Vd = 1, Vd2 = 1, Vp = 0.1, Vp2 = 0.1
param V0 = 5, V4 = 1, V7 = 1, V11e = 1, V11v = 1
param V15 = 1, V17 = 1, V19e = 1, V19v = 1

param k5 = 0.1, km5 = 0.1, k8 = 0.1, km8 = 0.1, k10 = 0.1, km10 = 0.1
param k12 = 1, km12 = 0.1, k13 = 1, km13 = 0.1, k20 = .1

param b0 = 0.001, b4 = 0.001, b7 = 0.001, b11 = 0.001
param b15 = 0.001, b17 = 0.001, b19 = 0.001

param d0 = 0.2, d1 = 0.4, d2 = 0.1, d3 = 0.2, d4 = 0.1, d5 = 0.1
param d7 = 0.1, d8 = 0.1, d9 = 0.01, d10 = 0.01,
param d11 = 0.1, d12 = 0.1, d13 = 0.01, d14 = 0.01
param d15 = 0.5, d16 = 0.1, d17 = 0.5, d18 = 0.1
param d19 = 0.5, d20 = 0.1, d21 = 10, d22 = 0.01, d23 = 0.01

param kin = 0.05, kout = 0.05
param pii = 1, DBT = 1, CYC = 1
```

A.3 Temperature Compensation

Here are the ODE files used for temperatures compensation in Chapter 5.

A.3.1 LG

```
# Goldbeter's model of circadian rhythms
# Leloup & Goldbeter, Chronobiology Intern. (1997)

param vs=0.76, vm=.65, Ki=1, Km=.5, n=4
param ks=.38, V1=3.2, K1=2, V2=1.58, K2=2
param V3=5, K3=2, V4=2.5, K4=2
param kin=1.9, kout=1.3, vd=.95, Kd=.2

PT = P0+P1+P2+Pn

init M=1.67, P0=0.622, P1=0.35, P2=0.22, Pn=0.34

dM/dt = vs/(1+(Pn/Ki)^n) - vm*M/(Km+M)
dP0/dt = ks*M - V1*P0/(K1+P0) + V2*P1/(K2+P1)
dP1/dt = V1*P0/(K1+P0) - V2*P1/(K2+P1) - V3*P1/(K3+P1) + V4*P2/(K4+P2)
dP2/dt = V3*P1/(K3+P1) - V4*P2/(K4+P2) -kin*P2 + kout*Pn - vd*P2/(Kd+P2)
dPn/dt = kin*P2 - kout*Pn
```

A.3.2 TH

```
# Simple PER Positive Feedback
# Tyson, Hong, Thron & Novak, Biophys. J. (1999)

param vm=1, km=0.1, vp=0.5, kp1=10, kp2=0.03, kp3=0.1
param Keq=200, Pcrit=0.1, Jp=0.05

q=2/(1+(1+8*Keq*Pt)^0.5)

dM/dt = vm/(1+(Pt*(1-q)/(2*Pcrit))^2) - km*M
dPt/dt = vp*M - (kp1*Pt*q+kp2*Pt)/(Jp+Pt) - kp3*Pt

init M=2.5,Pt=1
```

A.3.3 RS

```
# Simple PER Positive Feedback
# Hong, Conrad & Tyson, PNAS (2006)
#
# Same as Tyson, Hong, Thron & Novak Biophys J. (1999),
# modified to include the resetting hypothesis on parameter vp

param vm=2, km=0.2, kp1=53.36, kp2=0.06, kp3=0.2
param Keq=1, Pcrit=0.6, Jp=0.05
param mu=0.0288, sig=0.5, thresh=2

q=2/(1+sqrt(1+8*Keq*Pt))

dM/dt = vm/(1+(Pt*(1-q)/(2*Pcrit))^2) - km*M
dPt/dt = vp*M - (kp1*Pt*q+kp2*Pt)/(Jp+Pt) - kp3*Pt
dvp/dt = mu*vp

init M=2.5, Pt=1, vp=2

global -1 {Pt-thresh} {vp=sig*vp}
```

Appendix B

O8R Files

B.1 Optimization

Here are the O8R files used for optimization problems T1, T1-P, T2 and T3 in Chapter 3.

B.1.1 T1

```
<o8r>

  <!-- this sets the initial values of parameter and state variables -->
  <o8mi>
    <xdim>2</xdim>
    <species name="x" active="True" value="1" min="-1000" max="1000" />
    <species name="y" active="True" value="1" min="-1000" max="1000" />
    <pdim>4</pdim>
    <par name="p" active="True" value="1" min="-10" max="10" />
    <par name="a" active="True" value="-5" min="-10" max="10" />
    <par name="b" active="True" value="0.5" min="-10" max="10" />
    <par name="c" active="True" value="0.1" min="-10" max="10" />
    <vdim>0</vdim>
    <adim>0</adim>
  </o8mi>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>
    <sv name="x" />
    <par name="a" />

    <algorithm>3</algorithm><!-- 1 = vector, 2 = ~steepest decent, 3 = both 1 and 2 -->
    <perturbation_type>2</perturbation_type>
    <sigma_type>2</sigma_type>

    <sigma>0.01</sigma>
    <sigma_max>0.05</sigma_max>
    <sigma_min>0.01</sigma_min>

    <rho>0.01</rho>
    <rho_max>0.1</rho_max>
    <rho_min>0.0001</rho_min>
```

```
    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
    <n_best>10</n_best>\>
    <n_breaths>25</n_breaths>

    <smolder_tol>0.01</smolder_tol>
    <smolder_sigma_max>0.01</smolder_sigma_max>
    <smolder_sigma_min>0.0001</smolder_sigma_min>

    <!-- we want to match ST={(HB,-1.07,-0.25),(SN,-1.0,0),(SN,-1.1,0.45)} -->
    <match_map>
      <match_diagram>
        <type>2</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0.0005</qtol>
        <match_point><p_value>-1.07</p_value><x_value>-0.25</x_value><tau>3</tau></match_point>
        <match_point><p_value>-1</p_value><x_value>0</x_value><tau>2</tau></match_point>
        <match_point><p_value>-1.1</p_value><x_value>0.45</x_value><tau>2</tau></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

B.1.2 T1-P

```
<o8r>

  <!-- this sets the initial values of parameter and state variables -->
  <o8mi>
    <xdim>2</xdim>
    <species name="x" active="True" value="1" min="-1000" max="1000" />
    <species name="y" active="True" value="1" min="-1000" max="1000" />
    <pdim>4</pdim>
    <par name="p" active="False" value="1" min="-10" max="10" />
    <par name="a" active="True" value="-5" min="-10" max="10" />
    <par name="b" active="True" value="0.5" min="-10" max="10" />
    <par name="c" active="True" value="0.1" min="-10" max="10" />
    <vdim>0</vdim>
    <adim>0</adim>
  </o8mi>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>
    <sv name="x" />
    <par name="a" />

    <algorithm>3</algorithm><!-- 1 = vector, 2 = ~steepest decent, 3 = both 1 and 2 -->
    <perturbation_type>2</perturbation_type>
    <sigma_type>2</sigma_type>

    <sigma>0.01</sigma>
    <sigma_max>0.05</sigma_max>
    <sigma_min>0.01</sigma_min>
```

```
    <rho>0.01</rho>
    <rho_max>0.1</rho_max>
    <rho_min>0.0001</rho_min>

    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
    <n_best>10</n_best>\>
    <n_breaths>25</n_breaths>

    <smolder_tol>0.01</smolder_tol>
    <smolder_sigma_max>0.01</smolder_sigma_max>
    <smolder_sigma_min>0.0001</smolder_sigma_min>

    <!-- we want to match ST={(HB,-1.07,-0.25),(SN,-1.0,0),(SN,-1.1,0.45)} -->
    <match_map>
      <match_diagram>
        <type>2</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0.0005</qtol>
        <match_point><p_value>-1.07</p_value><x_value>-0.25</x_value><tau>3</tau></match_point>
        <match_point><p_value>-1</p_value><x_value>0</x_value><tau>2</tau></match_point>
        <match_point><p_value>-1.1</p_value><x_value>0.45</x_value><tau>2</tau></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

B.1.3 T1-R

```
<o8r>

  <!-- this sets the initial values of parameter and state variables -->
  <o8mi>
    <xdim>2</xdim>
    <species name="x" active="True" value="1" min="-1000" max="1000" />
    <species name="y" active="True" value="1" min="-1000" max="1000" />
    <pdim>4</pdim>
    <par name="p" active="True" value="1" min="-10" max="10" />
    <par name="a" active="True" value="-5" min="-10" max="10" />
    <par name="b" active="True" value="0.5" min="-10" max="10" />
    <par name="c" active="True" value="0.1" min="-10" max="10" />
    <vdim>0</vdim>
    <adim>0</adim>
  </o8mi>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>
    <sv name="x" />
    <par name="a" />

    <algorithm>3</algorithm><!-- 1 = vector, 2 = ~steepest decent, 3 = both 1 and 2 -->
    <perturbation_type>2</perturbation_type>
    <sigma_type>2</sigma_type>
```

```
    <sigma>0.01</sigma>
    <sigma_max>0.05</sigma_max>
    <sigma_min>0.01</sigma_min>

    <rho>0.01</rho>
    <rho_max>0.1</rho_max>
    <rho_min>0.0001</rho_min>

    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
    <n_best>10</n_best>\>
    <n_breaths>25</n_breaths>

    <smolder_tol>0.01</smolder_tol>
    <smolder_sigma_max>0.01</smolder_sigma_max>
    <smolder_sigma_min>0.0001</smolder_sigma_min>

    <!-- we want to match ST={(HB,-1.07,-0.25),(SN,-1.0,0),(SN,-1.1,0.45)} -->
    <match_map>
      <match_diagram>
        <type>3</type><!-- 3 is relative in p and x -->
        <penalty>100.0</penalty>
        <qtol>0.0005</qtol>
        <match_point><p_value>-1.07</p_value><x_value>-0.25</x_value><tau>3</tau></match_point>
        <match_point><p_value>-1</p_value><x_value>0</x_value><tau>2</tau></match_point>
        <match_point><p_value>-1.1</p_value><x_value>0.45</x_value><tau>2</tau></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

B.1.4 T2

```
<o8r>

  <!-- this sets the initial values of parameter and state variables -->
  <o8mi>
    <xdim>2</xdim>
    <species name="x" active="True" value="1" min="-1000" max="1000" type="1" />
    <species name="y" active="True" value="1" min="-1000" max="1000" type="1" />
    <pdim>4</pdim>
    <par name="p" active="True" value="1" min="-10" max="10" />
    <par name="a" active="True" value="-5" min="-10" max="10" />
    <par name="b" active="True" value="0.5" min="-10" max="10" />
    <par name="c" active="True" value="0.1" min="-10" max="10" />
    <vdim>0</vdim>
    <adim>0</adim>
  </o8mi>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>

    <sv name="x" />
    <par name="a" />
```

```
    <algorithm>3</algorithm><!-- 1 = vector, 2 = ~steepest decent, 3 = both 1 and 2 -->
    <perturbation_type>2</perturbation_type>
    <sigma_type>2</sigma_type>

    <sigma>0.25</sigma>
    <sigma_max>0.5</sigma_max>
    <sigma_min>0.0001</sigma_min>

    <rho>-0.05</rho>
    <rho_max>0.1</rho_max>
    <rho_min>0.0001</rho_min>

    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
    <n_best>10</n_best>\>
    <n_breaths>25</n_breaths>

    <smolder_tol>0.01</smolder_tol>
    <smolder_sigma_max>0.01</smolder_sigma_max>
    <smolder_sigma_min>0.0001</smolder_sigma_min>

    <!-- we want to match ST={(SN,0.61,0.95),(SN,-2.32,-0.6),(SN,-1.37,-1.1)},
         which incidentally will require b,c < 0 -->
    <match_map>
      <match_diagram><!-- here we have SN, SN, SN -->
        <type>2</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0.0005</qtol>
        <match_point><p_value>0.61</p_value><x_value>0.95</x_value><type>2</type></match_point>
        <match_point><p_value>-2.32</p_value><x_value>-0.6</x_value><type>2</type></match_point>
        <match_point><p_value>-1.37</p_value><x_value>-1.1</x_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- this is an intermediate, SN -->
        <type>2</type>
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>0.61</p_value><x_value>0.95</x_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- intermediate HB, HB -->
        <type>0</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-1</p_value><type>3</type></match_point>
        <match_point><p_value>-0.8</p_value><type>3</type></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

B.1.5 T2-R

```
<o8r>
```

```
<!-- this sets the initial values of parameter and state variables -->
<o8mi>
  <xdim>2</xdim>
  <species name="x" active="True" value="1" min="-1000" max="1000" type="1" />
  <species name="y" active="True" value="1" min="-1000" max="1000" type="1" />
  <pdim>4</pdim>
  <par name="p" active="True" value="1" min="-10" max="10" />
  <par name="a" active="True" value="-5" min="-10" max="10" />
  <par name="b" active="True" value="0.5" min="-10" max="10" />
  <par name="c" active="True" value="0.1" min="-10" max="10" />
  <vdim>0</vdim>
  <adim>0</adim>
</o8mi>

<!-- set up run to be a bifurcation match -->
<run name="opt" type="BifurcationMatch">
  <output>junk/opt</output>

  <sv name="x" />
  <par name="a" />

  <algorithm>3</algorithm><!-- 1 = vector, 2 = ~steepest decent, 3 = both 1 and 2 -->
  <perturbation_type>2</perturbation_type>
  <sigma_type>2</sigma_type>

  <sigma>0.5</sigma>
  <sigma_max>0.5</sigma_max>
  <sigma_min>0.0001</sigma_min>

  <rho>-0.05</rho>
  <rho_max>0.1</rho_max>
  <rho_min>0.0001</rho_min>

  <n_generations>1000</n_generations>
  <n_samples>10</n_samples>
  <n_best>10</n_best>\>
  <n_breaths>25</n_breaths>

  <smolder_tol>0.01</smolder_tol>
  <smolder_sigma_max>0.01</smolder_sigma_max>
  <smolder_sigma_min>0.0001</smolder_sigma_min>

  <!-- we want to match ST={(SN,0.61,0.95),(SN,-2.32,-0.6),(SN,-1.37,-1.1)},
       which incidentally will require b,c < 0 -->
  <match_map>
    <match_diagram><!-- here we have SN, SN, SN -->
      <type>3</type><!-- 0 = p only, 1 = relative, 2 = x data also, 3 = rel both -->
      <penalty>100.0</penalty>
      <qtol>0.0005</qtol>
      <match_point><p_value>0.61</p_value><x_value>0.95</x_value><type>2</type></match_point>
      <match_point><p_value>-2.32</p_value><x_value>-0.6</x_value><type>2</type></match_point>
      <match_point><p_value>-1.37</p_value><x_value>-1.1</x_value><type>2</type></match_point>
    </match_diagram>

    <match_diagram><!-- this is an intermediate, SN -->
      <type>3</type>
```

```
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>0.61</p_value><x_value>0.95</x_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- intermediate HB, HB -->
        <type>1</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-1</p_value><type>3</type></match_point>
        <match_point><p_value>-0.8</p_value><type>3</type></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

B.1.6 T3

```
<o8r>

  <!-- this sets the initial values of parameter and state variables (reverses t2 opt) -->
  <o8mi>
    <xdim>2</xdim>
    <species name="x" active="True" value="1" min="-1000" max="1000" type="1" />
    <species name="y" active="True" value="1" min="-1000" max="1000" type="1" />
    <pdim>4</pdim>
    <par name="p" active="True" value="1" min="-10" max="10" />
    <par name="a" active="True" value="-5" min="-5" max="5" />
    <par name="b" active="True" value="-0.5" min="-5" max="5" />
    <par name="c" active="True" value="-0.2" min="-1" max="1" />
    <vdim>0</vdim>
    <adim>0</adim>
  </o8mi>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>

    <sv name="x" />
    <par name="a" />

    <algorithm>3</algorithm><!-- 1 = vector, 2 = ~steepest decent, 3 = both 1 and 2 -->
    <perturbation_type>2</perturbation_type>
    <sigma_type>2</sigma_type>

    <sigma>0.25</sigma>
    <sigma_max>0.5</sigma_max>
    <sigma_min>0.0001</sigma_min>

    <rho>0.05</rho>
    <rho_max>0.1</rho_max>
    <rho_min>0.0001</rho_min>

    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
```

```
    <n_best>10</n_best>\>
    <n_breaths>25</n_breaths>

    <smolder_tol>0.01</smolder_tol>
    <smolder_sigma_max>0.01</smolder_sigma_max>
    <smolder_sigma_min>0.0001</smolder_sigma_min>

    <!-- we want to match ST={(HB,-0.65,-0.54),(SN,-0.12,0.22),(SN,-0.31,0.54)} -->
    <match_map>
      <match_diagram><!-- here we have HB, SN, SN -->
        <type>2</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0.0005</qtol>
        <match_point><p_value>-0.65</p_value><x_value>-0.54</x_value><type>3</type></match_point>
        <match_point><p_value>-0.12</p_value><x_value>0.22</x_value><type>2</type></match_point>
        <match_point><p_value>-0.31</p_value><x_value>0.54</x_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- this is an intermediate, SN -->
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>0.61</p_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- this is an intermediate, SN, SN -->
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-0.12</p_value><type>2</type></match_point>
        <match_point><p_value>-0.31</p_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- this is an intermediate, SN, SN, SN, don't care just hold on to it-->
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-0.12</p_value><type>2</type></match_point>
        <match_point><p_value>-0.12</p_value><type>2</type></match_point>
        <match_point><p_value>-0.31</p_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- intermediate HB, HB -->
        <type>0</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-65</p_value><type>3</type></match_point>
        <match_point><p_value>-0.31</p_value><type>3</type></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

B.1.7 T3-R

```
<o8r>

  <!-- this sets the initial values of parameter and state variables (reverses t2 opt) -->
  <o8mi>
    <xdim>2</xdim>
    <species name="x" active="True" value="1" min="-1000" max="1000" type="1" />
    <species name="y" active="True" value="1" min="-1000" max="1000" type="1" />
    <pdim>4</pdim>
    <par name="p" active="True" value="1" min="-10" max="10" />
    <par name="a" active="True" value="-5" min="-5" max="5" />
    <par name="b" active="True" value="-0.5" min="-5" max="5" />
    <par name="c" active="True" value="-0.2" min="-1" max="1" />
    <vdim>0</vdim>
    <adim>0</adim>
  </o8mi>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>

    <sv name="x" />
    <par name="a" />

    <algorithm>3</algorithm><!-- 1 = vector, 2 = ~steepest decent, 3 = both 1 and 2 -->
    <perturbation_type>2</perturbation_type>
    <sigma_type>2</sigma_type>

    <sigma>0.25</sigma>
    <sigma_max>0.5</sigma_max>
    <sigma_min>0.0001</sigma_min>

    <rho>0.05</rho>
    <rho_max>0.1</rho_max>
    <rho_min>0.0001</rho_min>

    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
    <n_best>10</n_best>\>
    <n_breaths>25</n_breaths>

    <smolder_tol>0.01</smolder_tol>
    <smolder_sigma_max>0.01</smolder_sigma_max>
    <smolder_sigma_min>0.0001</smolder_sigma_min>

    <!-- we want to match ST={(HB,-0.65,-0.54),(SN,-0.12,0.22),(SN,-0.31,0.54)} -->
    <match_map>
      <match_diagram><!-- here we have HB, SN, SN -->
        <type>3</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0.0005</qtol>
        <match_point><p_value>-0.65</p_value><x_value>-0.54</x_value><type>3</type></match_point>
        <match_point><p_value>-0.12</p_value><x_value>0.22</x_value><type>2</type></match_point>
        <match_point><p_value>-0.31</p_value><x_value>0.54</x_value><type>2</type></match_point>
      </match_diagram>
```

```
      <match_diagram><!-- this is an intermediate, SN -->
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>0.61</p_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- this is an intermediate, SN, SN -->
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-0.12</p_value><type>2</type></match_point>
        <match_point><p_value>-0.31</p_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- this is an intermediate, SN, SN, SN, don't care just hold on to it-->
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-0.12</p_value><type>2</type></match_point>
        <match_point><p_value>-0.12</p_value><type>2</type></match_point>
        <match_point><p_value>-0.31</p_value><type>2</type></match_point>
      </match_diagram>

      <match_diagram><!-- intermediate HB, HB -->
        <type>0</type><!-- 0 = p only, 1 = relative, 2 = x data also -->
        <penalty>100.0</penalty>
        <qtol>0</qtol>
        <match_point><p_value>-65</p_value><type>3</type></match_point>
        <match_point><p_value>-0.31</p_value><type>3</type></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

B.2 Circadian Rhythm Models

B.2.1 M0

T1

```
<o8r>

<o8mi>
  <xdim>4</xdim>
  <species name="Mp" active="True" value="0.1" min="0" max="100" />
  <species name="P0" active="True" value="1" min="0" max="1000" />
  <species name="P1" active="True" value="0" min="0" max="1000" />
  <species name="Pn" active="True" value="0" min="0" max="1000" />
  <pdim>17</pdim>
  <par name="n" active="True" value="2" min="0.00001" max="50" />
  <par name="DBT" active="False" value="1" min="0.00001" max="10" />
  <par name="PP2A" active="False" value="1" min="0.00001" max="10" />
  <par name="Jd" active="True" value="1" min="0.001" max="10" />
```

```
  <par name="Jp" active="True" value="1" min="0.001" max="10" />
  <par name="Ki" active="True" value="1" min="0.001" max="10" />
  <par name="d0" active="True" value="0.1" min="0" max="10" />
  <par name="d1" active="True" value="0.1" min="0" max="10" />
  <par name="d2" active="True" value="0.1" min="0" max="10" />
  <par name="d3" active="True" value="0.1" min="0" max="10" />
  <par name="kin" active="True" value="1" min="0.01" max="10" />
  <par name="kout" active="True" value="1" min="0.01" max="10" />
  <par name="b0" active="True" value="0.1" min="0" max="1" />
  <par name="s0" active="True" value="0.1" min="0" max="10" />
  <par name="Vd" active="True" value="1" min="0.001" max="10" />
  <par name="Vp" active="True" value="1" min="0.001" max="10" />
  <par name="V0" active="True" value="1" min="0.001" max="10" />
  <vdim>0</vdim>
  <adim>0</adim>
</o8mi>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>
    <sv name="P0" />
    <par name="n" />

    <algorithm init="2">3</algorithm>
    <perturbation_type>2</perturbation_type>

    <sigma type="2" min="0.001" max="0.5">0.05</sigma>
    <rho min="0.0001" max="0.1">-0.01</rho>
    <smolder sigma_min="0.0001" sigma_max="0.01">.25</smolder>

    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
    <n_best>10</n_best>\>

    <match_map>
      <match_diagram>
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>.2</qtol>
        <match_point><p_value>1</p_value><tau>3</tau></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

T2

```
<o8r>

  <o8mi>
    <xdim>4</xdim>
    <species name="Mp" active="True" value="0.1" min="0" max="100" />
    <species name="P0" active="True" value="1" min="0" max="1000" />
    <species name="P1" active="True" value=".5" min="0" max="1000" />
    <species name="Pn" active="True" value=".1" min="0" max="1000" />
    <pdim>17</pdim>
```

```
    <par name="n" active="True" value="2" min="0" max="100" />
    <par name="DBT" active="False" value="1" min="0.00001" max="10" />
    <par name="PP2A" active="False" value="1" min="0.00001" max="10" />
    <par name="Jd" active="True" value=".1" min="0.01" max=".2" />
    <par name="Jp" active="True" value=".01" min="0.01" max=".2" />
    <par name="Ki" active="True" value=".1" min="0.01" max=".2" />
    <par name="d0" active="True" value="0.5" min="0" max="10" />
    <par name="d1" active="True" value="0.25" min="0" max="10" />
    <par name="d2" active="True" value="0.25" min="0" max="10" />
    <par name="d3" active="True" value="0.25" min="0" max="10" />
    <par name="kin" active="True" value=".2" min="0.01" max=".2" />
    <par name="kout" active="True" value=".05" min="0.01" max=".2" />
    <par name="b0" active="True" value="0" min="0" max="1" />
    <par name="s0" active="True" value="1" min="0.001" max="10" />
    <par name="Vd" active="True" value=".75" min="0.1" max="2" />
    <par name="Vp" active="True" value=".4" min="0.1" max="2" />
    <par name="V0" active="True" value="1" min="0.1" max="2" />
    <vdim>0</vdim>
    <adim>0</adim>
  </o8mi>

  <continuation_pars>
    <nmx>10000</nmx>
    <ds>0.01</ds>
    <dsmin>0.01</dsmin>
  </continuation_pars>

  <!-- set up run to be a bifurcation match -->
  <run name="opt" type="BifurcationMatch">
    <output>junk/opt</output>
    <sv name="P0" />
    <par name="n" />

    <algorithm init="2">3</algorithm>
    <perturbation_type>2</perturbation_type>

    <sigma type="2" min="0.001" max="0.5">0.01</sigma>
    <rho min="0.0001" max="0.1">-0.01</rho>
    <smolder sigma_min="0.0001" sigma_max="0.01">1</smolder>

    <n_generations>1000</n_generations>
    <n_samples>10</n_samples>
    <n_best>10</n_best>\>

    <match_map>
      <match_diagram>
        <type>0</type>
        <penalty>100.0</penalty>
        <qtol>.25</qtol>
        <match_point><p_value>1</p_value><tau>3</tau></match_point>
      </match_diagram>
    </match_map>

  </run>
</o8r>
```

Bibliography

Bikem Akten, Eike Jauch, Ginka K. Genova, Eun Young Kim, Isaac Edery, Thomas Raabe, and F. Rob Jackson. A role for ck2 in the drosophila circadian oscillator. *Nature Neuroscience*, 6(3):251, 2003.

Lesley J. Ashmore and Amita Sehgal. A fly's eye view of circadian entrainment. *Journal Of Biological Rhythms*, 18(3):206, 2003.

S. Bao, J. Rihel, E. Bjes, J. Fan, and J.L. Price. The drosophila double-tiimes mutation delays the nuclear accumulation of period protein and affects the feedback regulation of period mrna. *The Journal of Neuroscience*, 21(18):7117–7126, 2001.

Mark T. Borisuk and John J. Tyson. Bifurcation analysis of a model of mitotic control in frog eggs. *Journal of Theoretical Biology*, 195(1):69–85, 1998.

Ania Busza, Myai Emery-Le, Michael Rosbash, and Patrick Emery. Roles of the two drosophila cryptochrome structural domains in circadian photoreception. *Science*, 304 (5676):1503, 2004.

J. M. Carlson and J. Doyle. Highly optimized tolerance: a mechanism for power laws in designed systems. *Physical Review. E, Statistical Physics, Plasmas, Fluids, And Related Interdisciplinary Topics*, 60(2, Part A):1412–1427, 1999.

J. M. Carlson and John Doyle. Complexity and robustness. *Proceedings Of The National Academy Of Sciences Of The United States Of America*, 99(Supplement 1):2538–2545, 2002.

Anthony R. Cashmore. Cryptochromes: enabling plants and animals to determine circadian time. *Cell*, 114(5):537–543, 2003.

Vijay Chickarmane, Sri R. Paladugu, Frank Bergmann, and Herbert M. Sauro. Bifurcation discovery tool. *Bioinformatics (Oxford, England)*, 21(18):3688–3690, 2005.

Emery D. Conrad and John J. Tyson. *Chapter 6: Modeling Molecular Interaction Networks With Nonlinear Ordinary Differential Equations*. Systems Modeling in Cellular Biology. MIT Press, 2006.

Shawn A. Cyran, Anna M. Buchsbaum, Karen L. Reddy, Meng-Chi Lin, Nicholas R. J. Glossop, Paul E. Hardin, Michael W. Young, Robert V. Storti, and Justin Blau. vrille, pdp1, and dclock form a second feedback loop in the drosophila circadian clock. *Cell*, 112 (3):329, 2003.

A. Dhooge, W. Govaerts, and Y. A. Kuznetsov. Matcont: A matlab package for numerical bifurcation analysis of odes. *ACM Trans. Math. Software*, 29:141–164, 2003.

Stephane Dissel, Veryan Codd, Robert Fedic, Karen J. Garner, Rodolfo Costa, Charalambos P. Kyriacou, and Ezio Rosato. A constitutively active cryptochrome in drosophila melanogaster. *Nature Neuroscience*, 7(8):834–840, 2004.

E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Y. A. Kuznetsov, B. Sandstede, and X. J. Wang. Auto97: Continuation and bifurcation software for ordinary differential equations (with homcont). user's guide. 1997.

P. Emery, W. V. So, M. Kaneko, J. C. Hall, and M. Rosbash. Cry, a drosophila clock and light-regulated cryptochrome, is a major contributor to circadian rhythm resetting and photosensitivity. *Cell*, 95(5):669, 1998.

Bard Ermentrout. *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*. SIAM, 2002.

Oren Froy, Dennis C. Chang, and Steven M. Reppert. Redox potential: Differential roles in dcry and mcry1 functions. *Current Biology*, 12(2):147, 2002.

Nicholas R. J. Glossop, Jerry H. Houl, Hao Zheng, Fanny S. Ng, Scott M. Dudek, and Paul E. Hardin. Vrille feeds back to control circadian transcription of clock in the drosophila circadian oscillator. *Neuron*, 37(2):249, 2003.

A. Goldbeter. A model for circadian oscillations in the drosophila period protein (per). *Proceedings Of The Royal Society Of London. Series B. Biological Sciences*, 261(1362): 319, 1995.

Didier Gonze, Jean-Christophe Leloup, and Albert Goldbeter. Theoretical models for circadian rhythms in neurospora and drosophila: Modeles theoriques pour les rythmes circadiens chez neurospora et chez la drosophile. *Comptes Rendus de l' Academie des Sciences - Series III - Sciences de la Vie*, 323(1):57, 2000.

Brian Goodwin. Oscillatory behavior in enzymatic control processes. *Advanced Enzyme Regulation*, 3:425–438, 1965.

Brigitte Grima, Annie Lamouroux, Elisabeth Chelot, Christian Papin, Bernadette Limbourg-Bouchon, and Francois Rouyer. The f-box protein slimb controls the levels of clock proteins period and timeless. *Nature*, 420(6912):178, 2002.

A. Gundel and M. B. Spencer. A circadian oscillator model based on empirical data. *Journal Of Biological Rhythms*, 14(6):516, 1999.

P. E. Hardin, J. C. Hall, and M. Rosbash. Feedback of the drosophila period gene product on circadian cycling of its messenger rna levels. *Nature*, 343(6258):536, 1990.

P. E. Hardin, J. C. Hall, and M. Rosbash. Circadian oscillations in period gene mrna levels are transcriptionally regulated. *Proceedings Of The National Academy Of Sciences Of The United States Of America*, 89(24):11711, 1992a.

P. E. Hardin, J. C. Hall, and M. Rosbash. Behavioral and molecular analyses suggest that circadian output is disrupted by disconnected mutants in d. melanogaster. *The EMBO Journal*, 11(1):1, 1992b.

Paul E. Hardin. The circadian timekeeping system of drosophila. *Current Biology*, 15(17): R714, 2005.

Charlotte Helfrich-Forster. The circadian system of drosophila melanogaster and its light input pathways. *Zoology*, 105(4):297, 2002.

Chris I. Hong, Emery D. Conrad, and John J. Tyson. A new hypothesis for temperature compensation in models of circadian rhythms. *submitted*, 2006.

M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, and al Cornish-Bowden et. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics (Oxford, England)*, 19(4):524–531, 2003.

V. Janssens and J. Goris. Protein phosphatase 2a: a highly regulated family of serine/threonine phosphatases implicated in cell growth and signalling. *The Biochemical Journal*, 353(Part 3):417, 2001.

Brian Kloss, Jeffrey L. Price, Lino Saez, Justin Blau, Adrian Rothenfluh, Cedric S. Wesley, and Michael W. Young. The drosophila clock gene double-time encodes a protein closely related to human casein kinase i[var epsilon]. *Cell*, 94(1):97, 1998.

Hyuk Wan Ko, Jin Jiang, and Isaac Edery. Role for slimb in the degradation of drosophila period protein phosphorylated by doubletime. *Nature*, 420(6916):673, 2002.

Ronald J. Konopka and Seymour Benzer. Clock mutants of drosophila melanogaster. *Proceedings of the National Academy of Sciences of the USA*, 68(9):2112–2116, 1971.

Yuri Kuznetsov. *Elements of Applied Bifurcation Analysis*, volume 12 (check this) of *Applied Mathematics*. Springer-Verlag, 2003.

Yuri Kuznetsov. Content-integrated environment for analysis of dynamical systems. tutorial. *Ecole Normale Superieure de Lyon, Rapport de Recherche UPMA*, 224, 1998.

J. C. Leloup and A. Goldbeter. Modeling the molecular regulatory mechanism of circadian rhythms in drosophila. *BioEssays: News And Reviews In Molecular, Cellular And Developmental Biology*, 22(1):84, 2000.

J. C. Leloup and A. Goldbeter. A molecular explanation for the long-term suppression of circadian rhythms by a single light pulse. *American Journal Of Physiology. Regulatory, Integrative And Comparative Physiology*, 280(4):R1206, 2001.

J. C. Leloup and A. Goldbeter. A model for circadian rhythms in drosophila incorporating the formation of a complex between the per and tim proteins. *Journal Of Biological Rhythms*, 13(1):70, 1998.

J. C. Leloup, D. Gonze, and A. Goldbeter. Limit cycle models for circadian rhythms based on transcriptional regulation in drosophila and neurospora. *Journal Of Biological Rhythms*, 14(6):433, 1999.

Jean-Christophe Leloup and Albert Goldbeter. Chaos and birhythmicity in a model for circadian oscillations of the per and tim proteins in drosophila. *Journal of Theoretical Biology*, 198(3):445, 1999.

Martin A. Lema, Diego A. Golombek, and Julian Echave. Delay model of the circadian pacemaker. *Journal of Theoretical Biology*, 204(4):565, 2000.

F. J. Lin, W. Song, E. Meyer-Bernstein, N. Naidoo, and A. Sehgal. Photic signaling by cryptochrome in the drosophila circadian system. *Molecular And Cellular Biology*, 21(21): 7287, 2001.

Jui-Ming Lin, Valerie L. Kilman, Kevin Keegan, Brie Paddock, Myai Emery-Le, Michael Rosbash, and Ravi Allada. A role for casein kinase 2alpha in the drosophila circadian clock. *Nature*, 420(6917):816, 2002.

Sebastian Martinek, Susan Inonog, Armen S. Manoukian, and Michael W. Young. A role for the segment polarity gene shaggy/gsk-3 in the drosophila circadian clock. *Cell*, 105(6): 769, 2001.

L. W. Morgan and J. F. Feldman. Epistatic and synergistic interactions between circadian clock mutations in neurospora crassa. *Genetics*, 159(2):537–543, 2001.

M. P. Myers, K. Wager-Smith, A. Rothenfluh-Hilfiker, and M. W. Young. Light-induced degradation of timeless and entrainment of the drosophila circadian clock. *Science*, 271 (5256):1736, 1996.

N. Naidoo, W. Song, M. Hunter-Ensor, and A. Sehgal. A role for the proteasome in the light response of the timeless clock protein. *Science*, 285(5434):1737, 1999.

Pipat Nawathean and Michael Rosbash. The doubletime and ckii kinases collaborate to potentiate drosophila per transcriptional repressor activity. *Molecular Cell*, 13(2):213, 2004.

Jeffrey L. Price, Justin Blau, Adrian Rothenfluh, Marla Abodeely, Brian Kloss, and Michael W. Young. double-time is a novel drosophila clock gene that regulates period protein accumulation. *Cell*, 94(1):83, 1998.

A. Rothenfluh, M. Abodeely, J. L. Price, and M. W. Young. Isolation and analysis of six timeless alleles that cause short- or long-period circadian rhythms in drosophila. *Genetics*, 156(2):665, 2000a.

Adrian Rothenfluh, Michael W. Young, and Lino Saez. A timeless-independent function for period proteins in the drosophila clock. *Neuron*, 26(2):505, 2000b.

P Ruoff, L Rensing, R Kommedal, and S Mohsenzadeh. Modeling temperature compensation in chemical and biological oscillators. *Chronobiol. Int.*, 14:499–510, 1997.

Sriram Sathyanarayanan, Xiangzhong Zheng, Rui Xiao, and Amita Sehgal. Posttranslational regulation of drosophila period protein by protein phosphatase 2a. *Cell*, 116(4):603, 2004.

Herbert M. Sauro, Michael Hucka, Andrew Finney, Cameron Wellock, Hamid Bolouri, John Doyle, and Hiroaki Kitano. Next generation simulation tools: the systems biology workbench and biospice integration. *Omics*, 7(4):355–372, 2003.

T. Scheper, D. Klinkenberg, C. Pennartz, and J. van Pelt. A mathematical model for the intracellular circadian rhythm generator. *The Journal Of Neuroscience: The Official Journal Of The Society For Neuroscience*, 19(1):40, 1999a.

T. O. Scheper, D. Klinkenberg, J. van Pelt, and C. Pennartz. A model of molecular circadian clocks: multiple mechanisms for phase shifting and a requirement for strong nonlinear interactions. *Journal Of Biological Rhythms*, 14(3):213, 1999b.

A. Sehgal, J. L. Price, B. Man, and M. W. Young. Loss of circadian behavioral rhythms and per rna oscillations in the drosophila mutant timeless. *Science*, 263(5153):1603, 1994.

Orie T. Shafer, Michael Rosbash, and James W. Truman. Sequential nuclear accumulation of the clock proteins period and timeless in the pacemaker neurons of drosophila melanogaster. *The Journal Of Neuroscience: The Official Journal Of The Society For Neuroscience*, 22(14):5946, 2002.

P. Smolen, D. A. Baxter, and J. H. Byrne. Modeling circadian oscillations with interlocking positive and negative feedback loops. *The Journal Of Neuroscience*, 21(17):6644, 2001.

Paul Smolen, Douglas A. Baxter, and John H. Byrne. A reduced model clarifies the role of feedback loops and time delays in the drosophila circadian oscillator. *Biophysical Journal*, 83(5):2349, 2002.

Paul Smolen, Paul E. Hardin, Brian S. Lo, Douglas A. Baxter, and John H. Byrne. Simulation of drosophila circadian oscillations, mutations, and light responses by a model with vri, pdp-1, and clk. *Biophysical Journal*, 86(5):2786, 2004.

W. V. So and M. Rosbash. Post-transcriptional regulation contributes to drosophila clock gene mrna cycling. *The EMBO Journal*, 16(23):7146, 1997.

Ralf Stanewsky, Maki Kaneko, Patrick Emery, Bonnie Beretta, Karen Wager-Smith, Steve A. Kay, Michael Rosbash, and Jeffrey C. Hall. The cryb mutation identifies cryptochrome as a circadian photoreceptor in drosophila. *Cell*, 95(5):681, 1998.

John J. Tyson, Chris I. Hong, C. Dennis Thron, and Bela Novak. A simple model of circadian rhythms based on dimerization and proteolysis of per and tim. *Biophysical Journal*, 77: 2411–2417, 1999.

E. Vielhaber, E. Eide, A. Rivers, Z. H. Gao, and D. M. Virshup. Nuclear entry of the circadian regulator mper1 is controlled by mammalian casein kinase 1 epsilon. *Molecular And Cellular Biology*, 20(13):4888, 2000.

L. B. Vosshall, J. L. Price, A. Sehgal, L. Saez, and M. W. Young. Block in nuclear localization of period protein by a second clock mutation, timeless. *Science*, 263(5153):1606, 1994.

H. Zeng, Z. Qian, M. P. Myers, and M. Rosbash. A light-entrainment mechanism for the drosophila circadian clock. *Nature*, 380(6570):129, 1996.

Jason W. Zwolak, John J. Tyson, and Layne T. Watson. Parameter estimation for a mathematical model of the cell cycle in frog eggs. *Journal of Computational Biology*, 12(1):48, 2005.

Vita

Contact Information

Emery D. Conrad
5076 Derring Hall
Department of Mathematics
Virginia Tech
Blacksburg, VA 24061-0406 USA

Voice: (540) 250-5151
Fax: (540) 231-9307
E-mail: econrad@vt.edu
Web: http://www.oscill8.com

Research Interests

Mathematical biology, computational cell biology, dynamical systems, bifurcation analysis, parameter estimation and optimization, biochemical regulatory networks, circadian rhythms

Education

Virginia Polytechnic Institute and State University (Virginia Tech)
Blacksburg, Virgina USA

Ph.D., Mathematics, May 2006

- Thesis Topic: "Bifurcation Analysis and Qualitative Optimization of Models in Molecular Cell Biology with Applications to the Circadian Clock"
- Advisor: John J. Tyson

M.S., Mathematics, May 1999

- Thesis Topic: "Mathematical Models of Biochemical Oscillation"
- Advisor: John J. Tyson

B.S., Mathematics, May 1998

- Undergraduate Research in Computational Algebra (*with Ed Green*)
- Minor: Russian

Honors and Awards

- Hatcher Scholarship, Mathematics Department, 1995/96, 1996/97
- Cobb Award for Excellence in the Russian Language, 1998
- Graduated Magna Cum Laude, 1998
- Phi Beta Kappa Honor Society, 1998

- Pi Mu Epsilon, Math Honor Society, 1998

Publications

- Emery D. Conrad, "Mathematical Models of Biochemical Oscillation," May 1999, Master's Thesis, `http://scholar.lib.vt.edu/theses/available/etd-051499-113229`.
- Emery D. Conrad, John J. Tyson, "Oscill8: A Dynamical Systems and Bifurcation Theory Toolset," August 2005, Conference Proceedings, Foundations of Systems Biology in Engineering (FOSBE 2005).
- Emery D. Conrad, John J. Tyson, April 2006. "Modeling Molecular Interaction Networks With Nonlinear Ordinary Differential Equations," Chapter 6 of *Systems Modeling in Cellular Biology*, MIT Press.
- Emery D. Conrad, John J. Tyson, "Oscill8: Tools and Techniques for Advanced Bifurcation Analysis and Parameter Optimization in Nonlinear ODEs, User Documentation," `http://oscill8.sourceforge.net/doc`.
- Christian I. Hong*, Emery D. Conrad*, John J. Tyson, "A New Hypothesis for Temperature Compensation in Models of Circadian Rhythms," *Submitted, first two authors contributed equally.*
- Mohsen Sabouri, Emery D. Conrad, Adrian Sandu, John Tyson, "A Simple Stochastic Model of Cell Cycle Regulation," *In preparation.*
- Tongli Zhang, Emery D. Conrad, Christian I. Hong, John J. Tyson, "Modeling Cyanobacterial Circadian Rhythm," *In preparation.*

Publications in Preparation from Dissertation

- Emery D. Conrad, John J. Tyson, "Oscill8: Tools and Techniques for Advanced Bifurcation Analysis and Parameter Optimization in Models of Biochemical Regulatory Networks."
- Emery D. Conrad, Mohsen Sabouri, John J. Tyson, "Incorporating Bifurcation Structure into a Parameter Optimization Scheme."
- Emery D. Conrad, John J. Tyson, "Positive Feedback in a Model of *Drosophila*'s Circadian Clock: The Role of PP2A."

Presentations

- Talk at University of Michigan Math Department, Bauer Center and Systems Biology Group at Harvard, "Bifurcation Analysis and Parameter Optimization: Techniques and Tools for Modeling Biochemical Regulatory Networks," 02/2006.
- Talk at Virginia Tech, SIAM Student Chapter Seminar Series, "Oscill8: Bifurcation Analysis and Parameter Optimization in Biological Models," 09/2005.
- Talk at SIAM Annual Conference, New Orleans, "Bifurcation Matching and Oscill8: Techniques and Tools for Model Identification," 07/2005.
- Talk at Keck Graduate School, Alameda California, "Tools for Advanced Bifurcation Analysis" 10/2004.

- Thesis defense at Virginia Tech, "Mathematical Models of Biochemical Oscillation," 05/1999.
- Virginia Tech (with Karen Potanka, Steve Sinnott, Mike Uy), "AGWAY: A Tool for Computational Algebra Over Finite Fields and the Berlekamp Factorization Algorithm," 05/1996.
- Mitre Corporation, "XSignOut: Implementation of a Multiuser Software Signout Board," 08/1993.

Posters

- Foundations of Systems Biology and Engineering (FOSBE) conference and BioSPICE PI Meeting, "Oscill8: A Dynamical Systems Toolset," 08/2005 and 11/2005.
- BioSPICE PI Meetings, "Oscill8: User Friendly Tools for Bifurcation Analysis," 10/2004, 05/2005.
- Society for Research on Biological Rhythms (SRBR) semi-annual conference, "Modeling Interlocking Loops in the Molecular Mechanism Underlying Circadian Rhythms," 06/2004.
- SRBR Conference, "Positive and Negative Feedback in the Molecular Mechanism Underlying Circadian Rhythms," 06/2002.

Teaching Experience

Virgina Tech, Blacksburg, Virginia USA

- Honors Tutor
  Responsibilities: One-on-one tutoring of all undergraduate math courses.
- Remedial Pre-Calculus
  Responsibilities: Created homework assignments, quizzes and tests, conducted lectures in tandem with partially pre-taped video lectures.
- Elementary Calculus with Trigonometry
  Responsibilities: Full teaching responsibilities (created lesson plans, quizzes and tests and lectured). This was a special course designed for Biology majors, and examples focused on biological applications.
- Vector Geometry for Engineers
  Responsibilities: Created lesson plans and quizzes, conducted lectures, and graded lab assignments. This was a special course which required students to use Mathematica in groups to solve weekly lab projects.

Florida State University, Tallahassee, Florida USA

Pre-Calculus
Responsibilities: Full teaching responsibilities (created lesson plans, quizzes and tests).

Grants

- DARPA BioSPICE, add-on to Grant #F30602-02-0572 (J. Tyson, P.I.), $100K, *Oscill8* development, 2005.

- National Security Education Program (NSEP) Award, $12K, Fall 1997 to Spring 1998.

Service

Faculty Search Committee for Computational Biologist, Virginia Tech, 2004/05, 2005/06.
Refereed paper, Bioinformatics, 2005.
Refereed papers, IEE Proceedings Systems Biology, 2005.

Memberships

American Mathematical Society (AMS).
Society for Industrial and Applied Mathematics (SIAM).

International

Languages

English (native), Serbian/Croatian (fluent), Russian (once fluent), courses in German, Hebrew, Japanese.

Study Abroad

Budapest, Hungary

- Junior Fellow, Collegium Budapest Institute for Advanced Study, Focus Group on Computational Biology, June 2003.

Moscow and St. Petersburg, Russia

- Moscow State University, Moscow, June to August 1997.
- Moscow International University, Moscow, August to December 1997.
- Herzen Pedagogical University, St. Petersburg, January to June 1998.

Professional and Industry Experience

- **Virgina Tech**, Blacksburg, Virginia USA

  Systems Architect II **January, 2005 - Present**

  I received a grant from DARPA as part of the BioSPICE project to complete a software tool used for the analysis of dynamical systems, Oscill8.

- **E. Conrad & Associates, LLC**, Blacksburg, Virginia USA

  Consultant **August, 2001 - Present**

  – Signal processing, AV compression, digital watermarking and cinema, compliance testing for MPEG, ATSC and PSIP.

  – Clients include the Sarnoff Corporation and the Virginia Tech Transportation Institute (VTTI).

- **Sarnoff Corporation**, Princeton, New Jersey USA

  Associate Member of the Technical Staff **January, 2000 - August, 2001**

  - Designed algorithms and implemented software for manipulation, compression and testing of imagery in the digital domain.
  - Supported digital cinema project and Sarnoff's Emmy Award-winning product line of Compliance Bitstreams (`http://www.sarnoff.com/BitStreams`).

- **Software Consulting** **Spring, 1997**
  - Enhanced the capabilities of *Statistica* for spatial econometrics.
  - Learned the relevant details of spatial econometrics and of the proprietary scripting language.

- **Mitre Corporation**, Virgina USA **May, 1993 - August, 1996**
  - Worked at CAASD/FAA developing algorithms and software for air traffic control.
  - Became proficient in Unix/Solaris, C, C++, Perl etc.

Computer Skills

- C, C++, C#, .NET/Mono, Java, GDI+, Tcl, Perl, Awk, PHP, CGI, HTML.
- CVS, Subversion, Perforce.
- MPEG1/2 audio and video, MPEG2 transport, Dolby.
- ATSC, PSIP, DVB-ASI, satellite.
- MATLAB, *Mathematica*, Maple, *Statistica*.
- LaTeX, GIMP, TpX, PowerPoint, Beamer-latex, GnuPlot.
- Operating systems: Unix, Linux, Solaris, Win2000, WinXP, MacOS X.