UNIVERSITY OF CALIFORNIA, SAN DIEGO

# Parameter And State Estimation In Nonlinear Dynamical Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Physics

by

Daniel R. Creveling

Committee in charge:

Professor Henry D. I. Abarbanel, Chair
Professor Daniel Arovas
Professor Philip Gill
Professor Thomas O'Neil
Professor Rick Salmon

2008

TABLE OF CONTENTS

# LIST OF FIGURES

LIST OF TABLES

## ACKNOWLEDGEMENTS

I would like to thank my adviser Henry Abarbanel for his insights, patience, approachability and support; without him this work would not have been possible. Also a big thank you goes to the research scientists, graduate students and staff of the Institute for Nonlinear Science for all of their support and intellectual stimulation over the years. I would like to especially thank Dr. Thomas Nowotny for introducing me to the Comedi open source data acquisition libraries and to rock climbing in Joshua Tree National Park. My graduate work at UCSD began with the non–neutral plasma group and I would like to thank Prof. Thomas O'Neil and Prof. Fred Driscoll for their support and scientific interaction during my first few years as a graduate student and also for their understanding and encouragement of my desire to study nonlinear science. I thought I was good in a laboratory coming from an engineering background and industry experience, but Prof. Fred Driscoll and Dr. Francois Anderegg showed me what first class experimentalists are really like! The time I spent working with them and learning from them will be cherished forever. I would also like to thank my good friend Dr. Eric Bass for many thoughtful scientific discussions and exciting mountaineering experiences!

Many people outside of UCSD offered support in many ways and I would like to thank some of them here. Stuart Downs for encouraging me to leave Northrop Grumman to follow my dreams, no matter how crazy they may be, and for many wonderful and stimulating conversations that somehow always included a bottle of wine. Don Auten, martial arts master instructor, extraordinary guitar player, and a wonderful friend who I could count on at all times. Kicking and punching the heavy bag or wrestling on the mat during weekends was an excellent way to prepare for another week of physics! Dr. Bruce McFarland who has known me since my EE days at Georgia Tech and has been a very supportive friend throughout my graduate school career. Pamela Rief whom I do not know how to thank enough for her patience, compassion and decreased entropy in my life. Lastly, I would like to thank my family, especially my father John and mother Peggy, for their continued love and support throughout all of the crazy adventures my life has conjured up.

## VITA

| | |
|---|---|
| 1997 | Bachelor of Electrical Engineering<br>Georgia Institute of Technology, Atlanta, Georgia |
| 2000 | Master of Science in Electrical Engineering<br>Georgia Institute of Technology, Atlanta, Georgia |
| 2000–2003 | Systems Engineer, Northrop Grumman<br>San Diego, California |
| 2004–2008 | Senior Electrical Engineer, SensorMetrix<br>San Diego, California |
| 2008 | Doctor of Philosophy in Physics<br>University of California, San Diego |

## PUBLICATIONS

Creveling, D. R., J. M. Jeanne, and H. D. I. Abarbanel, "Parameter Estimation using Balanced Synchronization" *Physics Letters A* **372**, 2043-2047 (2008).

Creveling, D. R., P. E. Gill, and H. D. I. Abarbanel, "State and Parameter Estimation in Nonlinear Systems as an Optimal Tracking Problem" *Physics Letters A* **372**, 2640-2644 (2008).

H. D. I. Abarbanel, Creveling, D. R., J. M. Jeanne, "Estimation of Parameters in Nonlinear Systems using Balanced Synchronization" *Physical Review E*, **77**, 016208 (2008).

Kamen E, Goldstein A, Creveling D, Sahinci E, Xiong Z, "Analysis of factors affecting component placement accuracy in SMT electronics assembly" [Conference Paper] 23$^{\text{rd}}$ IEEE/CPMT International Electronics Manufacturing Technology Symposium, pp. 50-7, 1998.

ABSTRACT OF THE DISSERTATION

**Parameter And State Estimation In Nonlinear Dynamical Systems**

by

Daniel R. Creveling
Doctor of Philosophy in Physics

University of California San Diego, 2008

Professor Henry D. I. Abarbanel, Chair

This thesis is concerned with the problem of state and parameter estimation in nonlinear systems. The need to evaluate unknown parameters in models of nonlinear physical, biophysical and engineering systems occurs throughout the development of phenomenological or reduced models of dynamics. When verifying and validating these models, it is important to incorporate information from observations in an efficient manner. Using the idea of synchronization of nonlinear dynamical systems, this thesis develops a framework for presenting data to a candidate model of a physical process in a way that makes efficient use of the measured data while allowing for estimation of the unknown parameters in the model.

The approach presented here builds on existing work that uses synchronization as a tool for parameter estimation. Some critical issues of stability in that work are addressed and a practical framework is developed for overcoming these difficulties. The central issue is the choice of coupling strength between the model and data. If the coupling is too strong, the model will reproduce the measured data regardless of the adequacy of the model or correctness of the parameters. If the coupling is too weak, nonlinearities in the dynamics could lead to complex dynamics rendering any cost function comparing the model to the data inadequate for the determination of model parameters. Two methods are introduced which seek to balance the need for coupling with the desire to allow the model to evolve in its natural manner without coupling. One method, 'balanced' synchronization, adds to the synchronization cost function a requirement that the conditional Lyapunov exponents of the model system, conditioned on being driven by the data, remain negative but small in magnitude. Another method allows the coupling between the data and the model to vary in time according to a specific form of differential equation. The coupling dynamics is damped to allow for a tendency toward zero coupling and driven by the synchronization error to increase coupling when needed. This method, along with a suitable cost function, allows for the determination of model parameters without the complexity of calculating Lyapunov exponents. Lastly, a method is developed allowing the coupling to vary in time without the constraint of following a differential equation. This approach shows the equivalence of the parameter and state estimation problem to that of tracking within an optimal control framework. This equivalence allows the application of powerful numerical methods that provide robust practical tools for model development and validation. Examples of each of these methods are presented with both simulated data and data measured from electrical circuit implementations of several dynamical systems.

# Chapter 1

# Introduction

## 1.1 Parameter Estimation in Physical Science

Estimating parameters in a nonlinear dynamical model, given observed data, is an important aspect of developing predictive models of physical and biological systems [27, 26, 8, 24, 28, 30, 39, 40, 18, 13]. Setting aside the issues of noise and experimental errors in acquiring the data and errors in the models themselves, one still has a significant challenge in estimating these parameters, especially when the dynamical behavior may be chaotic. As will be shown, the presence of positive Lyapunov exponents in this setting means that the numerical evaluation of a cost function representing parameter estimation quality, often a least squares metric, may suffer from sensitive dependence on initial conditions [1, 15].

Many methods have been explored for parameter estimation in nonlinear systems; two interesting ones are discussed in detail in [40]. The multiple shooting and extended Kalman filter approaches considered there show good results when applied to simple systems. The methods described in this thesis are meant to augment the toolbox of state and parameter estimation techniques and provide distinct advantages in some cases. For example, the method of optimal tracking discussed in Chapter 5 is preferred when it is desired to compare several candidate models against each other to determine which one describes the dynamics of a physical system more precisely. This method reformulates the problem of state and parameter estimation as an optimal tracking problem. The data is fed back to the model through a time dependent proportional error gain. Tools from optimal control theory are then used to solve for the optimal time dependent gain function. This function quantifies how much forcing is required for a model to track data from the system under study and may be used to compare the ability of a model to track measured data, allowing for a quantitative means of comparison between models. As another example, the method of dynamical coupling of data to a model presented in Chapters 3 is preferred when a quick, real–time comparison between data and a candidate model is desired. In this method the cost function is designed to force a trade–off between synchronization error and too much forcing though feedback coupling. A method of allowing the coupling gain to be self–adjusting is introduced and an interesting application of this case is presented in Chapter 4.

As stated above, the methods presented in this thesis are meant to augment the current methods used for parameter and state estimation. The methods herein are developed from the beginning with the complexities of nonlinear systems in mind. This ground up approach has resulted in powerful new methods of state and parameter estimation with advantages over those developed initially from linear system theory. Further, as these methods are applied to more complex and larger problems, their simplicity and efficiency provides an attractive alternative to the methods investigated in [40].

# Chapter 2

# Background Material

## 2.1 Overview of Example Systems

Several physical systems need to be defined for the purpose of facilitating discussion and for demonstrating the techniques of parameter and state estimation developed in this work. The systems of equations used here are the Lorenz system, the Colpitts system, and the Hodgkin-Huxley spiking neuron system. These systems are described in detail below.

### 2.1.1 Lorenz System

Edward Lorenz derived what have become known as the Lorenz equations [21] while studying atmospheric convection as a meteorologist at MIT. The phenomenon is driven by sunlight warming the ground causing heating to the lower layers of the atmosphere. This heating produces an upward flow of warm air and a corresponding downward flow of the more dense, cold air above. Lorenz began with the convection equations of Saltzman [31].

$$\begin{aligned}\frac{\partial}{\partial t}\nabla^2\psi &= -\frac{\partial(\psi,\nabla^2\psi)}{\partial(x,z)}+\nu\nabla^4\psi+g\alpha\frac{\partial\theta}{\partial x}\\ \frac{\partial}{\partial t}\theta &= -\frac{\partial(\psi,\theta)}{\partial(x,z)}+\frac{\Delta T}{H}\frac{\partial\psi}{\partial x}+\kappa\nabla^2\theta\end{aligned}\tag{2.1}$$

Where $\psi$ is the stream function for the motion and $\theta$ is the temperature deviation from steady state. The constants $g, \alpha, \nu, \kappa, H, \Delta T$ are respectively the gravitational acceleration, thermal expansion coefficient, kinematic viscosity, thermal conductivity, depth of the fluid, and the temperature difference between the upper and lower boundaries. Both $\psi$ and $\nabla^2\psi$ are taken to vanish at the upper and lower boundaries. The notation $\frac{\partial(a,b)}{\partial(x,z)}$ stands for the operator $\left(\frac{\partial a}{\partial x}\frac{\partial b}{\partial z} - \frac{\partial b}{\partial x}\frac{\partial a}{\partial z}\right)$. The functions $\psi$ and $\theta$ may be expanded in a Fourier series in $x$ and $z$ with time dependent coefficients to obtain a set of ordinary differential equations. Lorenz showed that interesting irregular and apparently non–periodic phenomena may be captured my truncating the Fourier series discarding all but three of the coefficients, thus obtaining the Lorenz equations (2.2).

$$\begin{aligned} \dot{\mathrm{X}} &= \sigma(\mathrm{Y} - \mathrm{X}) \\ \dot{\mathrm{Y}} &= \mathrm{RX} - \mathrm{Y} - \mathrm{XZ} \\ \dot{\mathrm{Z}} &= \mathrm{XY} - \mathrm{bZ} \end{aligned} \tag{2.2}$$

In (2.2), X is proportional to the intensity of the convection, Y is proportional to the temperature difference between ascending and descending currents, and Z is proportional to the distortion of the vertical temperature profile from linear. The parameters of this model are the Prandtl number, $\sigma$, a normalized Rayleigh number, R, and the parameter, b, related to the size of the region. Because of the extreme truncation involved, the solutions of equations (2.2) will not in general resemble the solutions of (2.1). Still, the Lorenz system exhibits rich dynamical behavior and has been extensively studied in its own right, independent of it meteorological roots. The Lorenz system is used here as a source of time–series data with complex temporal dynamics. A sample time–series of the Lorenz system and the corresponding attractor is shown in Figure 2.1.

In some cases, data will be taken from measurements on an electrical circuit (Figure 2.2) designed to integrate the Lorenz equations of motion [7]. This circuit provides a source of measured data from a real system subject to measurement noise as well as systematic errors due to variability in the actual component values. Differences in the actual resistance of the resistors and the actual capacitance of the capacitors from their labeled values cause the exact values of the parameters

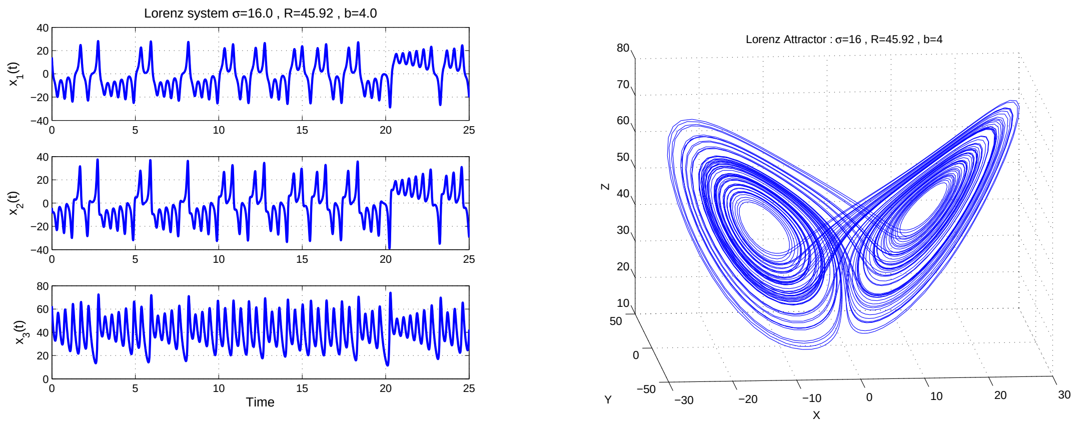


Figure 2.1: Lorenz system time–series and attractor

($\sigma$,R,b) to be unknown.

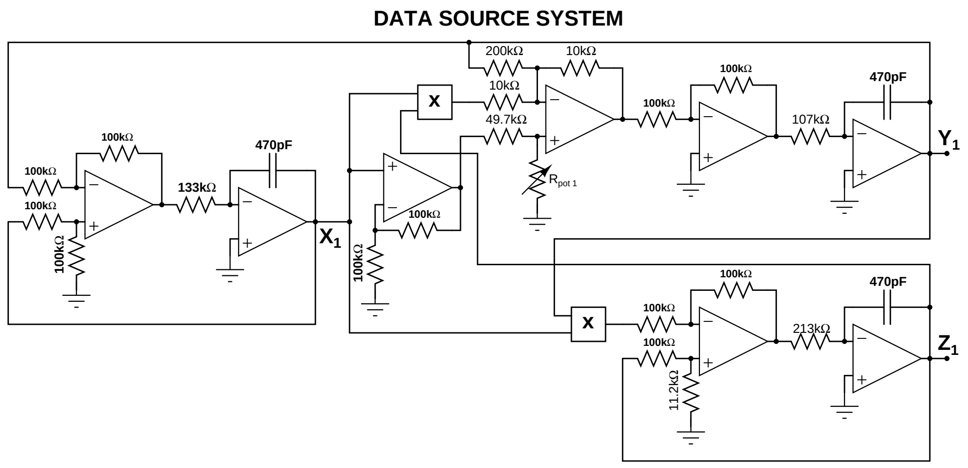


Figure 2.2: Lorenz system circuit

### 2.1.2 Colpitts Oscillator

The Colpitts oscillator is one member of a family of electronic oscillator circuits (Hartley, Clapp, Armstrong, etc.) and is shown in Figure 2.3. This simple electrical circuit consists of a single NPN bipolar junction transistor biased in its active region by $\pm V_o$ and $R_E$. The inductor L with series resistance $R_L$, and a capacitive divider composed of $C_1$ and $C_2$ provide the feedback network needed

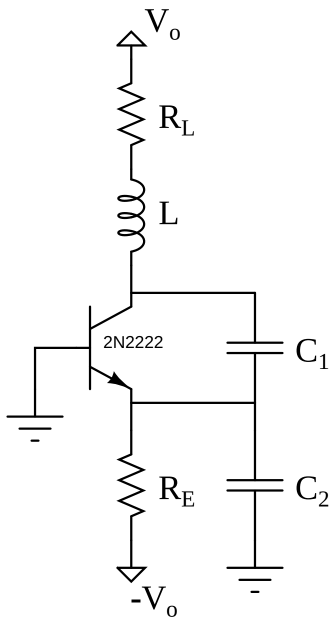


Figure 2.3: Colpitts oscillator circuit

for oscillation. It has a sinusoidal mode of operation with a wide frequency range from several hertz up to gigahertz without significant alteration to the design. The simplicity and robustness of the Colpitts oscillator has allowed for its widespread used in electronic devices and communication systems. Apart from the very useful sinusoidal mode of operation, this oscillator also exhibits complex dynamical behavior for a range of component values [16, 23]. The complexity of the dynamics is robust in the sense that it does not depend on the particular type of NPN transistor used. Many transistors of similar design (2N2222, 2N3904, BC108, etc.) will produce similar dynamical behavior. Therefore, the model of the circuit need not contain parameters specific to a particular model of NPN transistor (junction capacitance, Early voltage, etc.), the current gain $\beta$ and reverse–saturation current $\mathrm{I_S}$ are sufficient and we can use the Ebers-Moll [33] model for the transistor where $\mathrm{I_C} = \mathrm{I_S}(e^{(\mathrm{V_{BE}}/\mathrm{V_{th}})} - 1)$ and the current gain is defined by $\mathrm{I_C} = \beta \mathrm{I_B}$. $\mathrm{V_{th}}$ is the thermal voltage, $\mathrm{k_B T}/e$, and has a value at room temperature of $\mathrm{V_{th}} \approx 0.026\,\mathrm{Volts}$.

$$\begin{aligned}
\dot{\mathrm{V}}_{\mathrm{C1}} &= \frac{1}{\mathrm{C}_1}\left[\mathrm{I_L} - \mathrm{I_S}(e^{-\mathrm{V_{C2}}/\mathrm{V_{th}}} - 1)\right] \\
\dot{\mathrm{V}}_{\mathrm{C2}} &= \frac{1}{\mathrm{C}_2}\left[\mathrm{I_L} - (\mathrm{V_{C2}} + \mathrm{V_o})/\mathrm{R_E}\right] \\
\dot{\mathrm{I}}_{\mathrm{L}} &= \frac{1}{\mathrm{L}}\left[\mathrm{V_o} - \mathrm{R_L I_L} - \mathrm{V_{C2}} - \mathrm{V_{C1}}\right]
\end{aligned} \tag{2.3}$$

The analysis of the circuit in Figure 2.3, with some valid approximations, results in the above system of first order differential equations (2.3). $\mathrm{V_{C1}}$ is the voltage across capacitor $\mathrm{C_1}$ (the collector minus emitter voltage), $\mathrm{V_{C2}}$ is the voltage across capacitor $\mathrm{C_2}$ (the emitter voltage) and $\mathrm{I_L}$ is the current through the inductor L. There are several variations of this circuit each exhibiting complex dynamical behavior (replacing the emitter resistor with a current source for example). It is useful to cast the Colpitts system into a dimensionless form not connected with any particular circuit implementation but still maintaining the interesting dynamics. The common form of this generic Colpitts system is shown in equations (2.4) and contains the four parameters $\alpha$, $\gamma$, q, and $\eta$.

$$\begin{aligned}
\dot{x}_1 &= \alpha x_2 \\
\dot{x}_2 &= -\gamma(x_1 + x_3) - q x_2 \\
\dot{x}_3 &= \eta(x_2 + 1 - e^{-x1})
\end{aligned} \tag{2.4}$$

A sample time–series of the Colpitts system and the corresponding attractor is shown in Figure 2.4.

### 2.1.3 Hodgkin–Huxley Spiking Neuron Model

The Hodgkin–Huxley neuron model [12] is based on modeling the electrical properties of a patch of cell membrane by an equivalent circuit of the form shown in Figure 2.5. In the equivalent circuit, current flow across the membrane is divided into three distinct components, a sodium current $\mathrm{I}_{Na}$, a potassium current $\mathrm{I}_K$ and a small leakage current $\mathrm{I}_L$ containing all other ion species not explicitly modeled (mainly $\mathrm{Cl}^-$). The net current which flows into the cell through these channels has the effect of charging the membrane capacitance, giving the interior of the cell

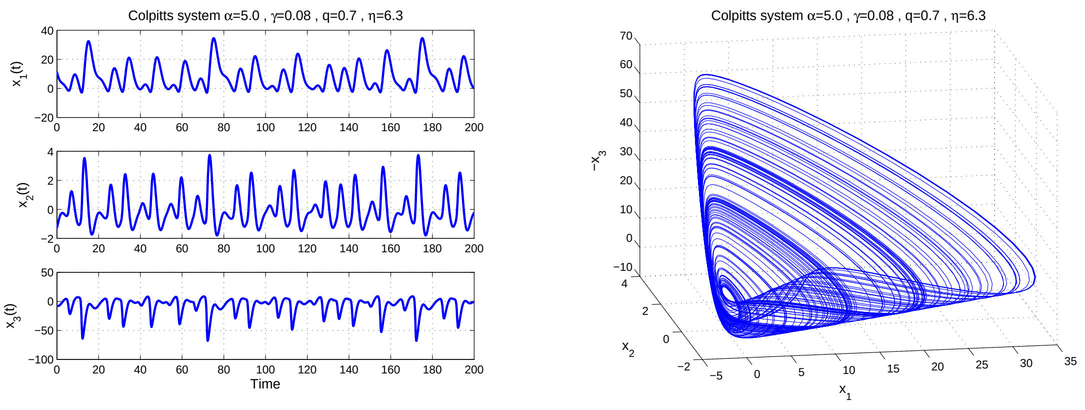


Figure 2.4: Colpitts oscillator time–series and attractor

a membrane potential V$_m$ relative to the exterior. The two variable conductances g$_K$ and g$_{Na}$ shown in the diagram represent the gating of potassium and sodium channels, and the constant leakage conductance g$_L$ represents the effect of other channels which are always open. Each of these channels is associated with an equilibrium potential represented by a battery in series with the conductance.

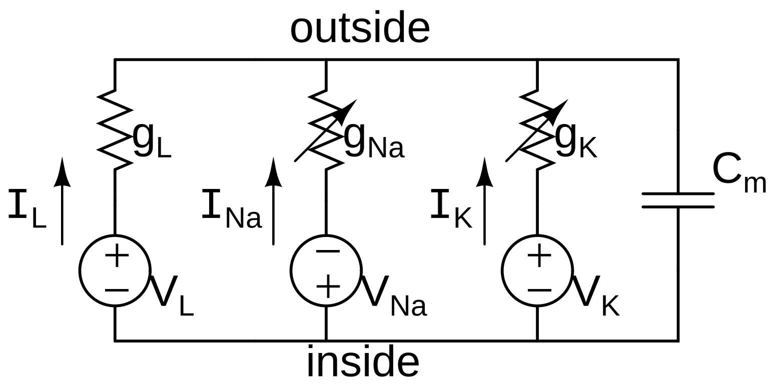


Figure 2.5: Hodgkin–Huxley model

Setting the net flow of current into the cell equal to the current charging the membrane capacitance gives the differential equation for V$_m$. This equation also includes an injected current I$_{inj}$ which allows for external excitation or inhibition of the cell. The variable conductances are themselves functions of n, m, and h. These are identified as the potassium activation variable, the sodium activation variable, and the sodium in–activation variable respectively [14]. The full system of differential equations is shown in equations (2.5) and a sample time–series for typical parameter values is shown in Figure 2.6.

$$\begin{aligned}
\dot{\mathrm{V}}_m &= \frac{1}{\mathrm{C}_m}\left[\tilde{g}_{Na}\mathrm{m}^3\mathrm{h}(V_{Na}-\mathrm{V}_m)+\tilde{g}_K\mathrm{n}^4(V_K-\mathrm{V}_m)+\tilde{g}_l(V_l-\mathrm{V}_m)+\mathrm{I}_{inj}\right]\\
\dot{\mathrm{n}} &= \alpha_n(\mathrm{V}_m)(1-\mathrm{n})-\beta_n(\mathrm{V}_m)\mathrm{n}\\
\dot{\mathrm{m}} &= \alpha_m(\mathrm{V}_m)(1-\mathrm{m})-\beta_m(\mathrm{V}_m)\mathrm{m}\\
\dot{\mathrm{h}} &= \alpha_h(\mathrm{V}_m)(1-\mathrm{h})-\beta_h(\mathrm{V}_m)\mathrm{h}
\end{aligned} \tag{2.5}$$

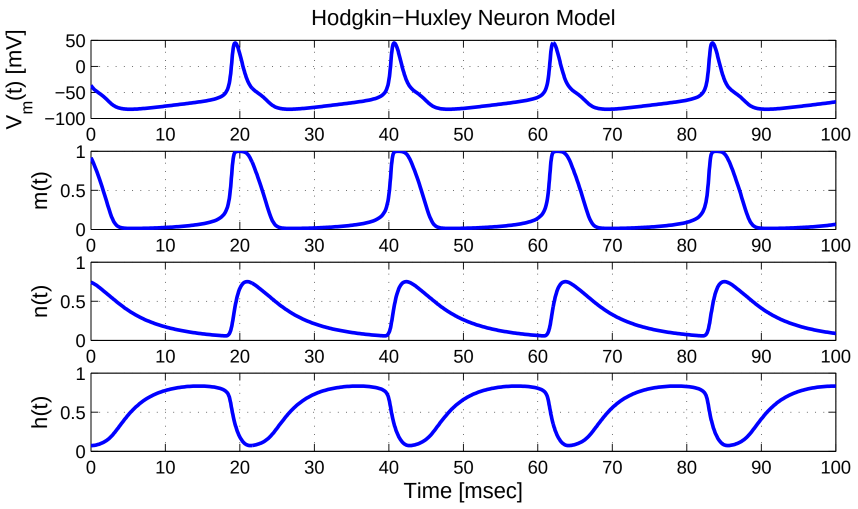


Figure 2.6: Hodgkin–Huxley time series

In this model, the rate coefficients for the gating variables n, m, and h are complex functions of the membrane potential $\mathrm{V}_m$ as shown in equations (2.6). Hodgkin and Huxley determined the form and estimated the parameters of these equations through many sets of electrophysiology experiments and numerical curve fitting. They were awarded the 1963 Nobel Prize in Physiology and Medicine (shared with john Eccles) for this work. In equations (2.6), the symbols $a_1, a_2, \ldots, a_8$ and $b_1, b_2, \ldots, b_7$ represent unknown parameters that need to be estimated before the model can make useful predictions. The techniques presented in this thesis will allow for the determination of these parameters from a time–series measurement of the membrane potential V$_m$.

$$
\begin{aligned}
\alpha_n(V_m) &= \frac{a_1(V_m+a_2)}{e^{a_3(V_m+a_2)}-1} & \beta_n(V_m) &= b_1 e^{b_2 V_m} \\
\alpha_m(V_m) &= \frac{a_4(V_m+a_5)}{e^{a_6(V_m+a_5)}-1} & \beta_m(V_m) &= b_3 e^{b_4 V_m} \\
\alpha_h(V_m) &= a_7 e^{a_8 V_m} & \beta_h(V_m) &= \frac{b_5}{e^{b_6(V_m+b_7)}+1}
\end{aligned} \tag{2.6}
$$

## 2.2 Dynamical Systems Overview

The example systems discussed above are all N–dimensional deterministic mathematical models of the form

$$
\begin{aligned}
\frac{dx_1}{dt} &= \mathrm{F}_1(x_1, x_2, \ldots, x_N, t, \mathbf{p}) \\
\frac{dx_2}{dt} &= \mathrm{F}_2(x_1, x_2, \ldots, x_N, t, \mathbf{p}) \\
&\vdots \\
\frac{dx_N}{dt} &= \mathrm{F}_N(x_1, x_2, \ldots, x_N, t, \mathbf{p})
\end{aligned}
$$

where $\mathbf{p}$ denotes all parameters associated with the system. This system of equations will often be written in vector form as $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}(t), t, \mathbf{p})$. The space denoted by $(x_1, x_2, \ldots, x_N)$ is referred to as phase space, and the path followed by an initial condition evolving through time is the phase space orbit or trajectory. It is typical of dissipative systems (systems with volume contracting regions in phase space) that the dynamics is characterized by an attractive set of points known as attractors. Typical attractors for the Lorenz and Colpitts systems are shown in Figure 2.1 and Figure 2.4 above. Any initial condition in the attractors basin of attraction will eventually be brought to the associated attractor and remain there. Although these attractors exist in a finite region of phase space, very complex dynamics may still occur. One measure of this complexity is contained in the Lyapunov exponents associated with the attractor. These exponents are associated with the stability of the attractive set and the ability to control their value is central to the parameter and state estimation scheme presented in this thesis.

### 2.2.1 Lyapunov Exponents

Lyapunov exponents characterize the evolution of small perturbations to an orbit. Consider a system following a fiducial trajectory denoted by $\mathbf{x}_*$ and a perturbed trajectory $\mathbf{x}_* + \eta$. The linearized dynamics provides an equation for the time evolution of small perturbations.

$$\begin{aligned}
\dot{\mathbf{x}}_* + \dot{\eta} &= \mathbf{F}(\mathbf{x}_* + \eta) \\
\dot{\mathbf{x}}_* + \dot{\eta} &= \mathbf{F}(\mathbf{x}_*) + \mathbf{DF}(\mathbf{x}_*) \cdot \eta + \cdots \\
\dot{\eta} &\approx \mathbf{DF}(\mathbf{x}_*) \cdot \eta
\end{aligned}$$

The linearized system characterizing the perturbation has exponential time dependence associated with the eigenvalues of the Jacobian matrix $\mathbf{DF}$. The eigenvalues of $\mathbf{DF}$ along a small orbit segment are known as local Lyapunov exponents and, when averaged over the entire attractor, converge to the associated Lyapunov exponents of the attractor. If the largest Lyapunov exponent is positive, a small perturbation will grow exponentially in time until it loses all association with its past neighboring points. This aspect is illustrated in Figure 2.7. The figure depicts the Lorenz attractor for $\sigma = 16$, R = 45.92 and b = 4.0 which has a largest Lyapunov exponent of $\lambda \approx 1.5$. The green circles represent a sphere of initial points centered on a point on the attractor. This sphere along with its center is iterated forward in time to produce the set of points represented by the red circles. As can be seen in Figure 2.8, the sphere is now distorted along two directions. One direction has grown, corresponding to the positive local Lyapunov exponent, and another direction has contracted corresponding to a negative local Lyapunov exponent. The sphere has kept its original radius in a third direction where the Lyapunov exponent is zero. This is characteristic of differential equations and represents a shift forward in time along the orbit. The yellow circles are the result of further integration forward in time where all memory of the initial sphere of points is lost. It is this property of nonlinear systems, positive lyapunov exponents, that gives rise to chaos and non–predictability. Small errors in initial conditions quickly give rise to large observable differences in the state. Any scheme for estimating states and parameters in chaotic systems must provide a means to overcome this inherent difficulty.

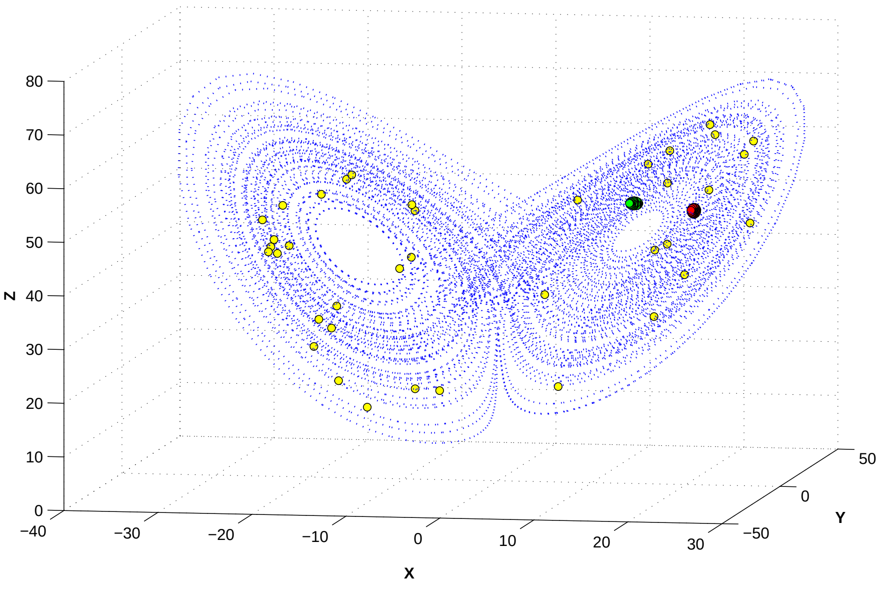


Figure 2.7: Illustration of Lyapunov Exponents

If only the largest Lyapunov exponent is desired, a simple method of calculation proceeds as follows [41, 34]. Choose two initial conditions separated by a small amount $|\delta \mathbf{r}_o| \sim 10^{-8}$ in any direction. Integrate these points forward in time N steps to get the new difference vector $\delta \mathbf{r}_N$. The choice of N should be such that the difference vector grows measurably while still remaining small enough for a linear approximation of the dynamics to hold. Store the quantity $\ln(\delta \mathbf{r}_o/\delta \mathbf{r}_N)$, re–normalize the difference vector to the original length and repeat the process iterating another N steps. The average of $\ln(\delta \mathbf{r}_o/\delta \mathbf{r}_N)$ from each iteration, divided by the time interval associated with N iteration steps, results in the largest Lyapunov exponent. Figure 2.9 demonstrates the convergence of this method for the Lorenz system with $\sigma = 16.0$, R $= 45.92$ and b $= 4.0$. A more sophisticated method [1] is required if all of the Lyapunov exponents of the system are required. In this case all of the eigenvalues of **DF** need to be calculated as the system evolves along the attractor.

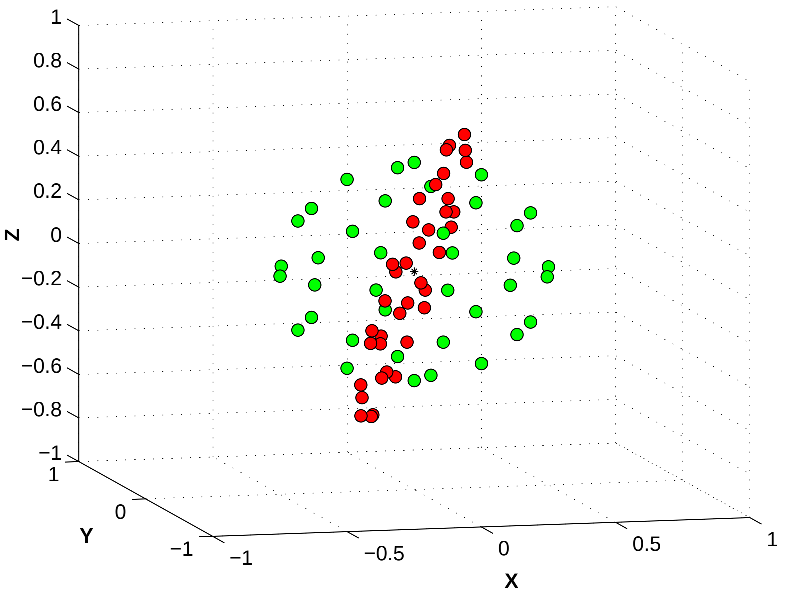


Figure 2.8: Distortion of initial sphere (green) to ellipsoid (red)

### 2.2.2 Synchronization

Central to the methods presented in this thesis is the ability of chaotic systems to synchronize with each other, either identically or in a more general fashion. Much published work exists on this subject [42, 3, 27, 29] and only the main point will be highlighted here. Consider two separate Lorenz systems evolving in time denoted by $\mathbf{X}(t)$ and $\mathbf{Y}(t)$. Owing to the chaos in the Lorenz system, even if these systems started with nearly identical initial conditions the corresponding time–series will have no correlation with each other at an arbitrary time in the future. If we require both chaotic systems to forever behave identically then we need a way to make the manifold $\mathbf{X}(t) = \mathbf{Y}(t)$ stable to small perturbations. One such strategy for accomplishing this goal is shown in equations (2.7).

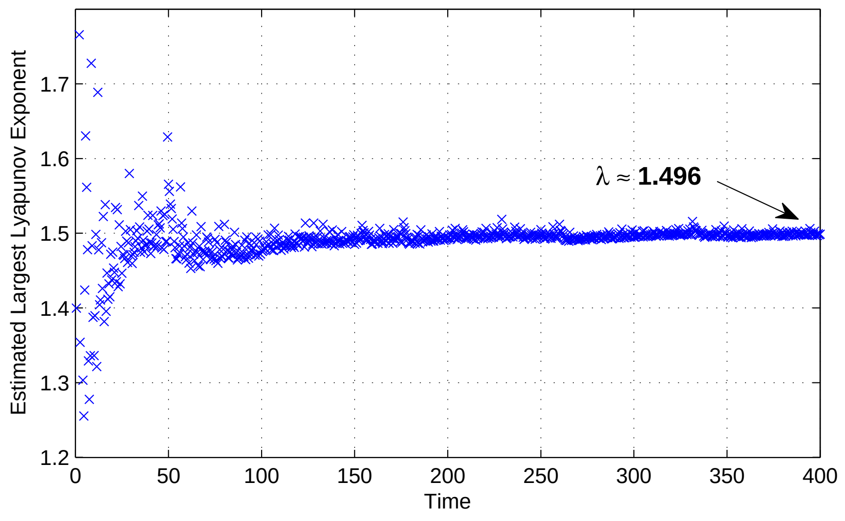


Figure 2.9: Largest Lyapunov Exponent calculation

$$\begin{aligned} \dot{\mathrm{x}}_1 &= \sigma(\mathrm{x}_2 - \mathrm{x}_1) & \dot{\mathrm{y}}_1 &= \sigma(\mathrm{y}_2 - \mathrm{y}_1) + k_{11}(\mathrm{x}_1 - \mathrm{y}_1) \\ \dot{\mathrm{x}}_2 &= \mathrm{R}\mathrm{x}_1 - \mathrm{x}_2 - \mathrm{x}_1\mathrm{x}_3 & \dot{\mathrm{y}}_2 &= \mathrm{R}\mathrm{y}_1 - \mathrm{y}_2 - \mathrm{y}_1\mathrm{y}_3 \\ \dot{\mathrm{x}}_3 &= \mathrm{x}_1\mathrm{x}_2 - \mathrm{b}\mathrm{x}_3 & \dot{\mathrm{y}}_3 &= \mathrm{y}_1\mathrm{y}_2 - \mathrm{b}\mathrm{y}_3 \end{aligned} \tag{2.7}$$

Data from the $\mathbf{X}(t)$ system is fed into the $\mathbf{Y}(t)$ system through a term proportional to the synchronization error $(\mathrm{x}_1(t) - \mathrm{y}_1(t))$. The Jacobian of the $\mathbf{Y}(t)$ system now becomes $[\mathbf{DF}(\mathbf{Y}(t)) - \mathbf{K}]$ where $\mathbf{K}$ is a matrix of coupling gains with $k_{ij}$ the coefficient for the $(\mathrm{x}_i(t) - \mathrm{y}_j(t))$ error term. In equations (2.7) only the $k_{11}$ term is non–zero. The associated Lyapunov exponents of this modified $\mathbf{Y}(t)$ system are called conditional Lyapunov exponents. The condition is that the exponents depend on the time–series $\mathrm{x}_1(t)$. For the synchronization manifold to be stable we need the largest conditional Lyapunov exponent to be negative. Figure 2.10 is a plot of the largest conditional Lyapunov exponent for error fed back into one, two and all three states. Note that as more states are fed back, the smaller the respective coupling gains are required to be for synchronization to occur. For single state feedback, a value of $k_{11} \approx 14$ is required for the synchronized state to be stable.

Figure 2.11 shows the time–series of both systems as the coupling strength takes on the three values $k_{11} = 0$, $k_{11} = 10$ and $k_{11} = 20$. After a small transient time, the synchronized stated $\mathbf{Y}(t) = \mathbf{X}(t)$ is reached for the value of coupling strength above the critical value of 14.

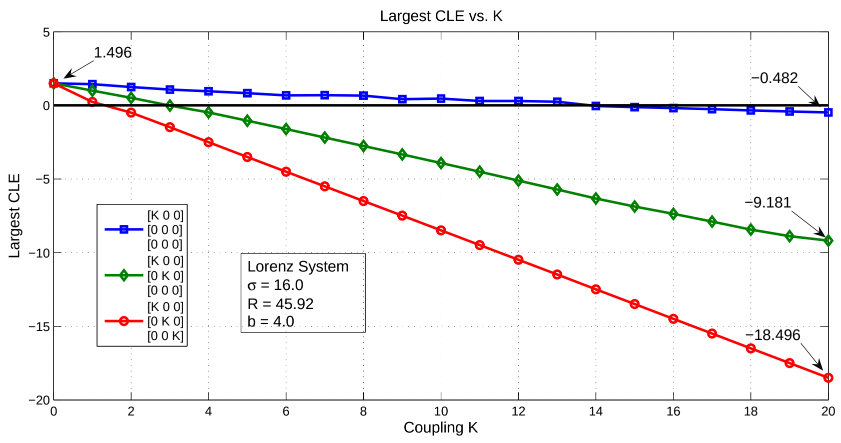


Figure 2.10: Largest Lyapunov exponent for Lorenz system with feedback

### 2.2.3 Phase Space Reconstruction

Here we use the idea of time delayed coordinates to reproduce the phase space of a dynamical system. Again taking the Lorenz system as the didactic example, we have a dynamical system consisting of the states $[x_1(t), x_2(t), x_3(t)]$ but we will assume we are limited to measuring only $x_1(t)$. One technique for studying the dynamics of a system given limited time–series measurement is to use time lagged variables for reconstruction of the phase space [1, 25, 32, 37]. We begin by constructing a d-dimensional vector of time lagged variables separated in time by T.

$$\mathbf{y}(n) = [x_1(n), x_1(n+T), x_1(n+2T), \ldots, x_1(n+(d-1)T)]$$

A sufficient condition on the dimension, d, is that it be an integer larger than twice the dimension of the attractor. This would guarantee the attractor in phase space and the reconstructed attractor in time–delayed space are related to each

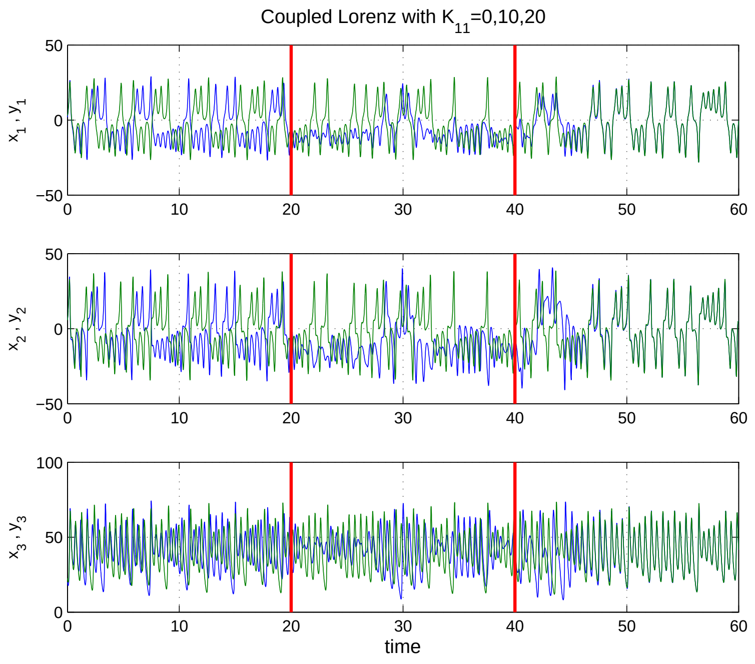


Figure 2.11: Lorenz system synchronization

other by a smooth nonlinear transformation. Physical properties of the attractor calculated in phase space may instead be calculated in the time–delayed space. As for determining the best value for the time delay T, in theory any value will work. In practice it is worth considering how much additional information is obtained by making a measurement at $x_1(n+T)$ after a measurement at $x_1(n)$. If the time delay is too small, the system will only have changed slightly and the vector of time delays will contain numbers too close in value to be of any practical use. One useful metric for determining the optimal time delay is a quantity borrowed from information theory: the average mutual information [10, 1]. The amount one learns about the measurement of $\hat{a}$ from a measurement of $\hat{b}$ is quantified as

$$\mathrm{I}_{AB}(\hat{a},\hat{b}) = \log\left[\frac{\mathrm{P}_{AB}(\hat{a},\hat{b})}{\mathrm{P}_A(\hat{a})\mathrm{P}_B(\hat{b})}\right]$$

where $\mathrm{P}_A(\hat{a})$ is the probability of observing $\hat{a}$ out of the set of all possible observations A, and $\mathrm{P}_B(\hat{b})$ is the probability of observing $\hat{b}$ out of the set of all possible observations B. $\mathrm{P}_{AB}(\hat{a},\hat{b})$ is the probability of observing both $\hat{a}$ and $\hat{b}$. The quan-

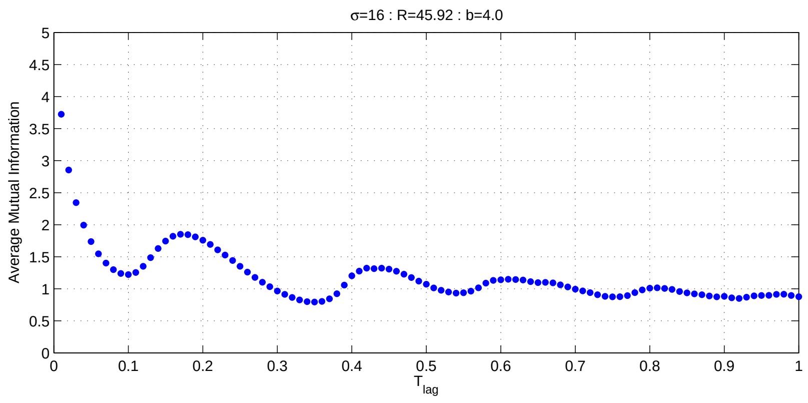


Figure 2.12: Average Mutual Information vs. Time Delay

tity $\mathrm{I}_{AB}(\hat{a}, \hat{b})$ is the mutual information between the two measurements $\hat{a}$ and $\hat{b}$. Averaging over all possible measurements provides the average mutual information between measurement $\hat{a}$ and $\hat{b}$. We use this concept to calculate the average mutual information between a measurement of the physical system at $x_1(n)$ and a measurement at $x_1(n+T)$.

$$\mathrm{I}(T) = \sum_{n=1}^{N} \mathrm{P}(x_1(n), x_1(n+T)) \log \left[ \frac{\mathrm{P}(x_1(n), x_1(n+T))}{\mathrm{P}(x_1(n))\mathrm{P}(x_1(n+T))} \right]$$

Figure 2.12 shows a plot of the average mutual information as a function of the time delay T for the Lorenz system and shows several clear minima. Given a measurement of $x_1(n)$ we know the least amount of information about a measurement of $x_1(n+0.1)$. We should therefore wait until $x_1(n+0.1)$ for the next measurement to be placed in the time delayed vector. A plot of the original $[x_1(t), x_2(t), x_3(t)]$ and reconstructed $[x_1(n), x_1(n+0.1), x_1(n+0.2)]$ phase space attractors is shown in Figure 2.13.

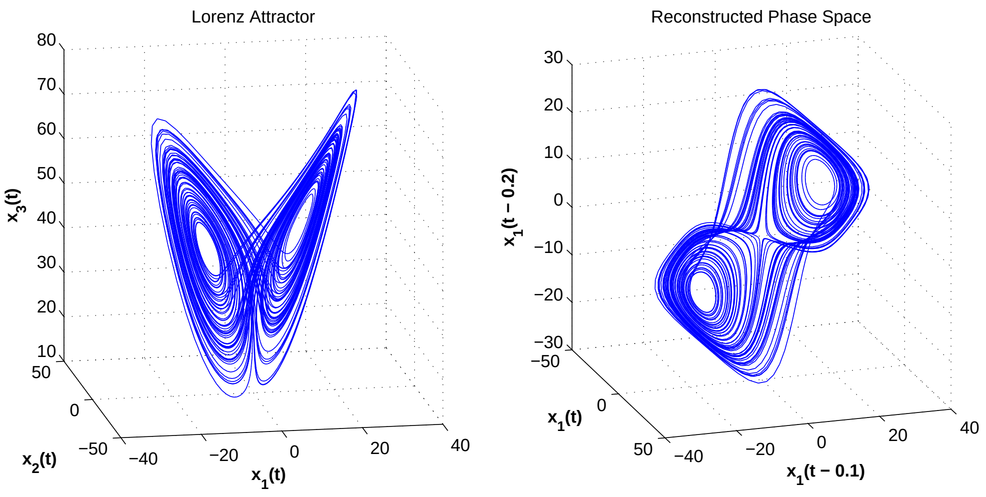


Figure 2.13: Lorenz attractors : Phase Space and Time Delay Space

# Chapter 3

# Parameter Estimation

## 3.1 Background and Description

The problem of state and parameter estimation may be cast into the following form. We imagine there is a physical or biological system whose state is determined by the N dimensional vector $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_N(t)] = [x_1(t), \mathbf{x}_\perp(t)]$ where measurements are made on the system at time intervals $\tau$. Starting with some initial time $t_0$, measurements are made at times $t_0 + m\tau\,;\, m = 0, 1, 2, \ldots, M$, resulting in observations $\mathbf{X}(m) = \mathbf{X}(t_0 + m\tau)$. It is assumed this sampling is adequate to capture the frequencies of importance in the operation of the system of interest: $\tau$ is small enough, and $M\tau$ is large enough. One component of the state is now measured and stored for use in determining parameters in a model describing the physical system of interest. This could be an arbitrary scalar function of the system's state $h(\mathbf{x}(t))$ but here we will take it as one of the state variables itself, namely, $x_1(t)$. The other N-1 state variables $\mathbf{u}(t) = [x_2(t), x_3(t), ..., x_N(t)]$ are unobserved. If more than one state is observed the analysis would proceed in a parallel fashion to what is presented below.

The observed (or 'driver') system satisfies differential equations in $\mathbf{x}(t)$ that depend on a fixed set of parameters $\mathbf{p}$, i.e.,

$$\begin{aligned}\frac{dx_1(t)}{dt} &= F_1(x_1(t), \mathbf{x}_\perp(t), \mathbf{p}) \\ \frac{d\mathbf{x}_\perp(t)}{dt} &= \mathbf{F}_\perp(x_1(t), \mathbf{x}_\perp(t), \mathbf{p}).\end{aligned}$$

where the vector fields $F_1(\bullet)$ and $\mathbf{F}_\perp(\bullet)$ are presumed known, and the trajectories $\mathbf{X}(t) = [x_1(t), \mathbf{x}_\perp(t)]$ are determined by the initial conditions and the $P$ parameters $\mathbf{p} = [p_1, p_2, ..., p_P]$.

The time series information $x_1(t)$ (the 'data') is now passed to the receiver system which is the model for the process describing the observations. To illustrate the methods we take this model to be precisely that used in generating the data. The dynamics are known but we assume we do not know the parameters $\mathbf{q} = [q_1, q_2, ..., q_P]$ of the receiver (the model). The state of the receiver is given by $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_N(t)] = [y_1(t), \mathbf{y}_\perp(t)]$. Without coupling to the data the model satisfies

$$\begin{aligned} \frac{dy_1(t)}{dt} &= F_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}) \\ \frac{d\mathbf{y}_\perp(t)}{dt} &= \mathbf{F}_\perp(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}), \end{aligned} \tag{3.1}$$

and we seek to determine the $\mathbf{q}$ given a time–series $x_1(t)$ [28]. For this purpose, we couple the receiver system to the input signal $x_1(t)$ using

$$\begin{aligned} \frac{dy_1(t)}{dt} &= F_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}) + K(x_1(t) - y_1(t)) \\ \frac{d\mathbf{y}_\perp(t)}{dt} &= \mathbf{F}_\perp(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}). \end{aligned} \tag{3.2}$$

Had the signal $h(\mathbf{x}(t))$ been measured, the coupling term would have been $K(h(\mathbf{x}(t)) - h(\mathbf{y}(t)))$ [28]. For some range of values of the scalar $K$, the model will synchronize to the data $\mathbf{y}(t) \approx \mathbf{x}(t)$, and the model will be "most accurate" when the model parameters realize $\mathbf{q} = \mathbf{p}$. The conditional Lyapunov exponent (CLE) [27] of the model system must be negative for synchronization to occur.

As a principle to assist in estimating $\mathbf{q}$, a natural choice is to minimize the cost function $C(\mathbf{q}) = \frac{1}{2M}\sum_{m=1}^{M} g((X_1(m) - Y_1(m))^2)$, where $g(z^2) \approx z^2$ for small $z$; $\mathbf{Y}(m) = \mathbf{y}(t_0 + m\tau)$. This minimization involves seeking a zero of

$$\frac{\partial C(\mathbf{q})}{\partial q_\alpha} = \int dt (y_1(t; \mathbf{q}) - x_1(t)) \frac{\partial y_1(t; \mathbf{q})}{\partial q_\alpha}, \tag{3.3}$$

for $\alpha = 1, 2, \ldots, P$. The $\mathbf{q}$ giving these zeros will be the estimate of the model parameters $\mathbf{q}$ needed to match the settings $\mathbf{p}$ of the data source. Formally it is also desired that the $P \times P$ matrix

$$\frac{\partial^2 C(\mathbf{q})}{\partial q_\alpha \partial q_\beta} \tag{3.4}$$

is positive definite so that a minimum has indeed been found.

The quantity

$$\frac{\partial y_1(t;\mathbf{q})}{\partial q_\alpha} \tag{3.5}$$

is determined, along with

$$\frac{\partial \mathbf{y}_\perp(t;\mathbf{q})}{\partial q_\alpha} \tag{3.6}$$

by

$$\frac{d}{dt}\begin{pmatrix} \frac{\partial y_1(t;\mathbf{q})}{\partial \mathbf{q}} \\ \frac{\partial \mathbf{y}_\perp(t;\mathbf{q})}{\partial \mathbf{q}} \end{pmatrix} = \left[\mathbf{DF}(y_1(t;\mathbf{q}), \mathbf{y}_\perp(t;\mathbf{q})) - \begin{pmatrix} K & 0 \\ 0 & 0 \end{pmatrix}\right]\begin{pmatrix} \frac{\partial y_1(t;\mathbf{q})}{\partial \mathbf{q}} \\ \frac{\partial \mathbf{y}_\perp(t;\mathbf{q})}{\partial \mathbf{q}} \end{pmatrix} + \frac{\partial \mathbf{F}(\mathbf{y},\mathbf{q})}{\partial \mathbf{q}}, \tag{3.7}$$

where $\mathbf{DF}$ is the $N \times N$ Jacobian matrix

$$\mathbf{DF}_{ij}(\mathbf{y}) = \frac{\partial F_i(\mathbf{y})}{\partial y_j}\,; i,j = 1,2,...,N \tag{3.8}$$

of the model dynamics, and the matrix involving $K$ is also $N \times N$ with only the upper left diagonal element nonzero. $\mathbf{F}(\mathbf{y})$ is the total N-dimensional vector field $\mathbf{F}(\mathbf{y}) = (F_1(y_1, \mathbf{y}_\perp), \mathbf{F}_\perp(y_1, \mathbf{y}_\perp))$.

The issue of stability needs to be addressed as the eigenvalues of the Jacobian $\mathbf{DF}(\mathbf{y})$ iterated along the orbit $\mathbf{y}(t)$ may have positive conditional Lyapunov exponents; conditioned on the driving signal $x_1(t)$. These may be found by concatenating products of the matrices $\mathbf{DF}(\mathbf{y}(t)) - \mathbf{K}$ and relying on the Oseledec theorem for the existence of the eigenvalues of the iterated product. If there are positive conditional Lyapunov exponents, then the synchronization manifold $y_1(t) = x_1(t)$ is not stable to small perturbations, and the evaluation of the derivatives of the cost function, Equation (3.3), is numerically uncertain [40, 1, 15]. The conditional Lyapunov exponents, however, can be made negative by increasing the magnitude of $K$. When the largest conditional Lyapunov exponent has become negative, then the synchronization manifold is stable to small perturbations, and evaluating the derivatives of the cost function is straightforward.

As $K$ becomes large, the term $K(x_1(t) - y_1(t))$ dominates the right hand side of the evolution equation for $y_1(t)$ unless $x_1(t) - y_1(t) \approx \frac{1}{K}$ or smaller. As this happens, all the derivatives in Equation (3.3) approach zero, and there is little numerical variation of the derivatives as functions of the parameters $\mathbf{q}$. The minimum of the cost function becomes so flat in $\mathbf{q}$ space it is numerically extremely difficult to locate. As we cure the numerical instability associated with chaos in the model dynamical system, we may be sent into a regime where the evaluation of the parameters $\mathbf{q}$ by minimizing the cost function becomes harder and harder. In addition, when $K$ is large, the dynamics of the model are entrained by the driving $K(x_1(t) - y_1(t))$ and any model is forced to follow $x_1(t)$. The ability to distinguish among models is therefore lost. A balance is required between these two unacceptable limits, and to accomplish that, a way is needed to choose a value of $K$ that leads to the largest conditional Lyapunov exponent being just negative, yet is not such a large value of $K$ that we lose the ability to see variations in the $\mathbf{q}$. This approach is called "balanced synchronization," and two ways to achieve this is explored below.

## 3.2 Controlling the Largest CLE

For synchronization of the experimental data $x_1(t)$ and the model system to be effective in estimating the parameters $\mathbf{q}$, the largest conditional Lyapunov exponent must be negative [27]. Since this method is dependent on synchronization to drive the model dynamical variables $[y_1(t), \mathbf{y}_\perp(t)]$ to those taken on by the observed signal $[x_1(t), \mathbf{x}_\perp(t)]$, if synchronization fails, the foundation of the method would fail [28]. The conditional Lyapunov exponents are evaluated by perturbing the receiving (or model) system from the synchronization manifold $\mathbf{x}(t) = \mathbf{y}(t)$. Linearizing the perturbed dynamics we have for $\Delta(t) = \mathbf{y}(t) - \mathbf{x}(t)$

$$\frac{d\Delta(t)}{dt} = [\mathbf{DF}(\mathbf{x}(t)) - \mathbf{K}] \cdot \Delta(t), \tag{3.9}$$

where

$$\mathbf{K} = \begin{pmatrix} K & 0 \\ 0 & 0 \end{pmatrix}. \tag{3.10}$$

Recalling that the data and the model are sampled at time intervals $\tau$, this differential equation may be interpreted as a map between values of the perturbation at 'time' $n$ and time $n+1$:

$$\Delta(n+1) = \mathbf{DH}(\mathbf{x}(n)) \cdot \Delta(n), \tag{3.11}$$

where

$$\mathbf{DH}(\mathbf{x}(n)) = \mathbf{I} + \tau[\mathbf{DF}(\mathbf{x}(n)) - \mathbf{K}], \tag{3.12}$$

and $\mathbf{I}$ is the unit $N \times N$ matrix.

To calculate all of the conditional Lyapunov exponents, this map needs to be iterated and the eigenvalues of the resulting product of matrices evaluated. However, all of the conditional Lyapunov exponents are not required, only the largest, and this entails a much easier calculation. Take an arbitrary unit vector $\mathbf{w}$ in the N-dimensional space and multiply it by the iterated 'effective' Jacobian

$$\mathbf{DH}(\mathbf{x})^L = \mathbf{DH}(\mathbf{x}(L)) \cdot \mathbf{DH}(\mathbf{x}(L-1)) \cdots \mathbf{DH}(\mathbf{x}(1)), \tag{3.13}$$

which carries the linearized perturbation at 'time' 1 to its value at 'time' L+1. Multiply the vector $\mathbf{DH}(\mathbf{x})^L\mathbf{w}$ by its transpose

$$\left(\mathbf{DH}(\mathbf{x})^L\mathbf{w}\right)^T \cdot \left(\mathbf{DH}(\mathbf{x})^L\mathbf{w}\right). \tag{3.14}$$

This grows as $e^{2L\lambda(\mathbf{q},K)}$ for large $L$. The quantity

$$\frac{1}{2L} \log \left(\mathbf{DH}(\mathbf{x})^L\mathbf{w}\right)^T \cdot \left(\mathbf{DH}(\mathbf{x})^L\mathbf{w}\right), \tag{3.15}$$

is just $\lambda(\mathbf{q}, K)$, the largest conditional Lyapunov exponent. This $\lambda(\mathbf{q}, K)$ is desired to be slightly negative [1].

This suggests replacing the least squares cost function with the balanced cost function

$$C(K, \mathbf{q}) = \frac{1}{2} \int dt f((x_1(t) - y_1(t; \mathbf{q}))^2) + \frac{1}{2}(\lambda(\mathbf{q}, K) + \xi)^2, \tag{3.16}$$

where $\xi$ is a small negative number and $f(x^2)$ a function which vanishes as $x^2$ near $x = 0$. This enforces synchronization by asking that the first term be small, but does not allow $K$ to be so large that $\lambda(\mathbf{q}, K)$ is too negative. As an approximation to this cost function for $K$ large, the first term is estimated as being of order $\frac{1}{K^2}$, as $x_1(t) - y_1(t, \mathbf{q}) \approx \frac{1}{K}$ while the second grows as $K^2$. If the total cost function is approximated as

$$\frac{A}{K^2} + BK^2, \tag{3.17}$$

with $A$ and $B$ constants, this has a minimum at approximately $K \approx (\frac{A}{B})^{1/4}$. So $K$ remains bounded, and there is a balance between the smallness of the synchronization error and the magnitude of the synchronization coupling strength $K$.

As an example consider the Lorenz system. There are three dynamical equations for the driving oscillator

$$\begin{aligned}\frac{dx_1(t)}{dt} &= \sigma(x_2(t) - x_1(t))\\ \frac{dx_2(t)}{dt} &= -x_2(t) + R_D x_1(t) - x_1(t)x_3(t)\\ \frac{dx_3(t)}{dt} &= = -bx_3(t) + x_1(t)x_2(t),\end{aligned}$$

and three equations for the driven receiver

$$\begin{aligned}\frac{dy_1(t)}{dt} &= \sigma(y_2(t) - y_1(t)) + K_{11}(x_1(t) - y_1(t))\\ \frac{dy_2(t)}{dt} &= -y_2(t) + Ry_1(t) - y_1(t)y_3(t)\\ \frac{dy_3(t)}{dt} &= = -by_3(t) + y_1(t)y_2(t).\end{aligned}$$

where conventional values for the parameters $\sigma = 16.0$ and $b = 4.0$ were chosen, and the driver oscillator had $R_D = 45.92$. As described above, a cost function is chosen which balances the least squares deviation of the driver input $x_1(t)$ and the model output $y_1(t)$ against the deviation of the largest CLE $\lambda(K_{11}, R)$ from a small negative number, Cost$(K_{11}, R)$, Equation(3.16) with $\xi = 0.05$, and varied $K_{11}$ and $R$. With $T = 5000$ samples of the driving Lorenz model trajectory and the driven trajectory we see in Figure (3.1) that there is a clear minimum in $\mathrm{C}_{\mathrm{min}}(R)$, the cost function minimum for a given $R$ as $K_{11}$ is varied, at $R \approx R_D$. In Figure (3.2) the entire cost function $\mathrm{C}(K_{11}, R)$ is plotted. In the latter there is again a clear minimum for $K_{11} \approx 14$ and at $R \approx R_D$. This minimum results from the balance in the cost function between the two terms – synchronization error and small negative

CLE. Were the second term absent, the cost function would simply flatten out as $K_{11}$ is increased, causing numerical difficulty in finding the minimum value.

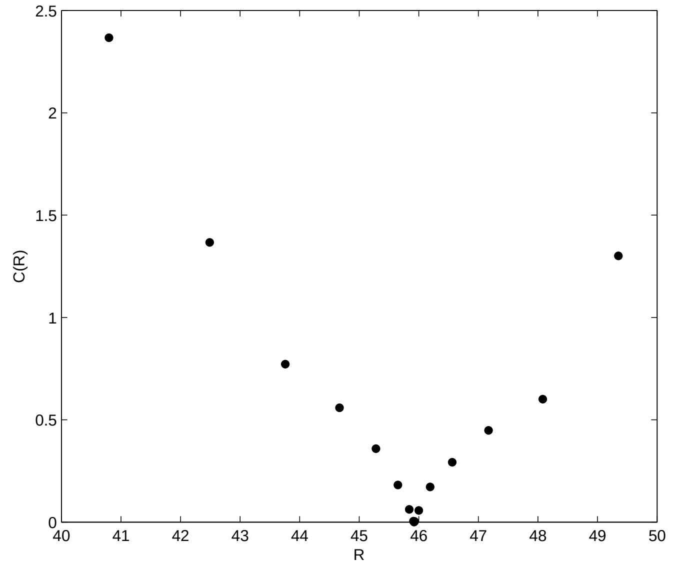


Figure 3.1: Lorenz : Cost vs. R

Figure 3.3 illustrates results from the three dimensional Colpitts oscillator. Selecting the parameter values $\gamma_D = 0.0797$, $q_D = 0.6898$ and $\eta_D = 6.2723$, data $x_1(t)$ is collected for $\alpha_D = 5.0$, a regime where the Colpitts oscillator exhibits chaotic behavior. This data is then used to drive a second Colpitts oscillator with the 'unknown' parameter $\gamma$ and choose the cost function, Cost$(K, \gamma)$, Equation(3.16), balancing the deviation from synchronization against the magnitude of the largest CLE with $\xi = 0.05$. Figure (3.3) shows Cost$(K, \gamma)$. There is a clear minimum establishing a value for $K \approx 0.5$ and indicating that $\gamma \approx 0.08$. If the CLE term were absent, there would be a minimum in the $\gamma$ variation for small $K$, and that would flatten out and numerically vanish as $K$ increased. The optimization principle, Equation (3.16), determines both a value for the model parameter $\gamma$ and a value for the coupling strength guaranteeing synchronization.

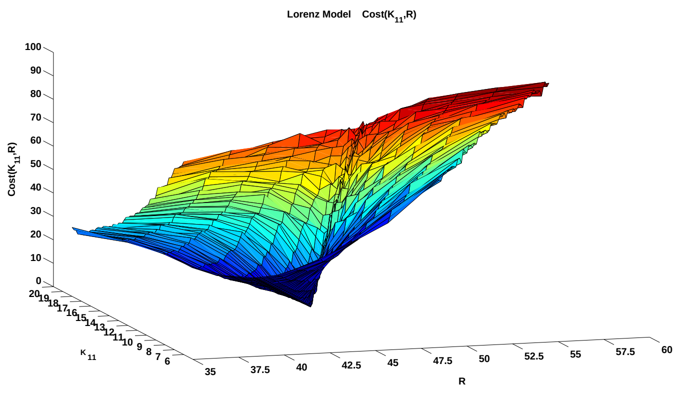


Figure 3.2: Lorenz : Cost(R,$K_{11}$)

## 3.3 Dynamical Coupling

In contrast to using the coupling strength as an additional parameter to be searched over, the coupling could itself be a dynamical variable designed such that only the minimal amount of information necessary to sustain synchronization is passed from the data generating system to the model system. When the model system does not match the observed data, information needs to flow to the model and the coupling should increase. When the data and model are in agreement the coupling should decrease to allow the model system to run more independently. The dynamical coupling method may be thought of as a means of controlling the flow of information from the data generating system to the model. An alternate interpretation is that of automatically adjusting the coupling to maintain a small but negative CLE. The design of the coupling dynamics and a cost function for estimating parameters is described below using the Lorenz system, as well as results from a physical implementation of the method using electrical circuits.

### 3.3.1 Description

Consider the following differential equations for $y_1(t), \mathbf{y}_\perp(t)$ and $K(t)$

$$\frac{dy_1(t)}{dt} = F_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}) + K(t)(x_1(t) - y_1(t))$$

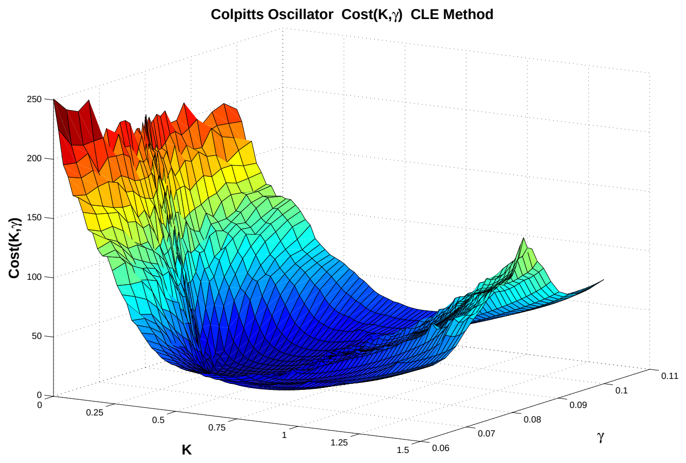


Figure 3.3: Colpitts : Cost$(\gamma, K_{11})$

$$\begin{aligned}\frac{d\mathbf{y}_\perp(t)}{dt} &= \mathbf{F}_\perp(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}) \\ \frac{dK(t)}{dt} &= -\alpha K(t) + g((\mathbf{x}(t) - \mathbf{y}(t))^2) \qquad (3.18)\end{aligned}$$

where $\alpha > 0$, $g(0) = 0$, and for small argument $g(x^2) \approx x^2$. The solution to the $K(t)$ differential equation

$$K(t) = e^{-at}K(t=0) + \int_0^t dt' e^{-a(t-t')} g((\mathbf{x}(t') - \mathbf{y}(t'))^2), \qquad (3.19)$$

shows that the initial value of $K(t = 0)$ is unimportant when $ta \gg 0$. If $g(x^2)$ is bounded by a constant $C$, $K(t) < \frac{C}{\alpha}$. This means that the coupling is again balanced, but this time by the requirements of synchronization on the magnitude of $g(x^2)$ and the tendency for $K$ to vanish. As time goes by there are intervals when the synchronization is lost and $(\mathbf{x}(t') - \mathbf{y}(t'))^2$ grows, this leads to a growth in the magnitude of $K(t)$ improving the synchronization. The balance between these two effects, decay of $K(t)$ to zero and growth of $K(t)$ to strengthen synchronization, is the embodiment of balanced synchronization in this method. For these investigations, two different functions of the synchronization error $x_1(t) - y_1(t)$ were explored

$$\frac{dK(t)}{dt} = -aK(t) + \tanh((\mathbf{x}(t) - \mathbf{y}(t))^2),$$

$$\frac{dK(t)}{dt} = -aK(t) + (\mathbf{x}(t) - \mathbf{y}(t))^2. \tag{3.20}$$

In the first $K(t) < \frac{1}{\alpha}$, while for the second the perturbation to $K(t)$ may become quite large.

This approach to balanced synchronization, can never have identity synchronization $\mathbf{x}(t) = \mathbf{y}(t)$, as the driving system (the data) is $N$ dimensional ($\mathbf{x}(t)$) while the model system is $N+1$ dimensional ($\mathbf{y}(t), K(t)$). If the coupling parameter $K(t)$ were constant, identity synchronization would be possible. In the formulation of a criterion as given below for determining the parameters $\mathbf{q}$, the notion of identity synchronization is still used, and this may be approximately correct as the variation of $K(t)$ is bounded. Nonetheless, as will be show below, the determination of parameters in the model using this method works very well. This is due to the fact that generalized synchronization between these two dynamical systems is possible [29, 2]. A test for this is to use the variant of the auxiliary system method as discussed by D. Tang [38]. In this approach the signal from the driving system $x_1(t)$ is repeatedly presented to the model system $\mathbf{y}(t), K(t)$, Equation(3.18). In each presentation the model system is taken to be in a different state by adjusting either the time at which $x_1(t)$ is presented or adjusting the initial conditions of the model system. In effect, the state of the model system is different for each presentation. If we call the model output from presentation $n = 1, 2, ..., ; \mathbf{y}^{(n)}(t)$, then after a transient, these outputs from the various realizations of the model should agree: $\mathbf{y}^{(n)}(t) = \mathbf{y}^{(n')}(t)\,; n \neq n'$. Results of such a test are shown in Figure 3.4.

## 3.3.2 Lorenz Example

$$\begin{aligned} \dot{\mathrm{x}}_1 &= \sigma_1(\mathrm{x}_2 - \mathrm{x}_1) \\ \dot{\mathrm{x}}_2 &= \mathrm{R}_1\mathrm{x}_1 - \mathrm{x}_2 - \mathrm{x}_1\mathrm{x}_3 \\ \dot{\mathrm{x}}_3 &= \mathrm{x}_1\mathrm{x}_2 - \mathrm{b}_1\mathrm{x}_3 \end{aligned} \qquad \begin{aligned} \dot{\mathrm{y}}_1 &= \sigma_2(\mathrm{y}_2 - \mathrm{y}_1) + K(t)(\mathrm{x}_1 - \mathrm{y}_1) \\ \dot{\mathrm{y}}_2 &= \mathrm{R}_2\mathrm{y}_1 - \mathrm{y}_2 - \mathrm{y}_1\mathrm{y}_3 \\ \dot{\mathrm{y}}_3 &= \mathrm{y}_1\mathrm{y}_2 - \mathrm{b}_2\mathrm{y}_3 \\ \dot{K} &= -\alpha K(t) + 4\tanh\left[(\frac{x_1 - y_1}{\gamma})^2\right] \end{aligned} \tag{3.21}$$

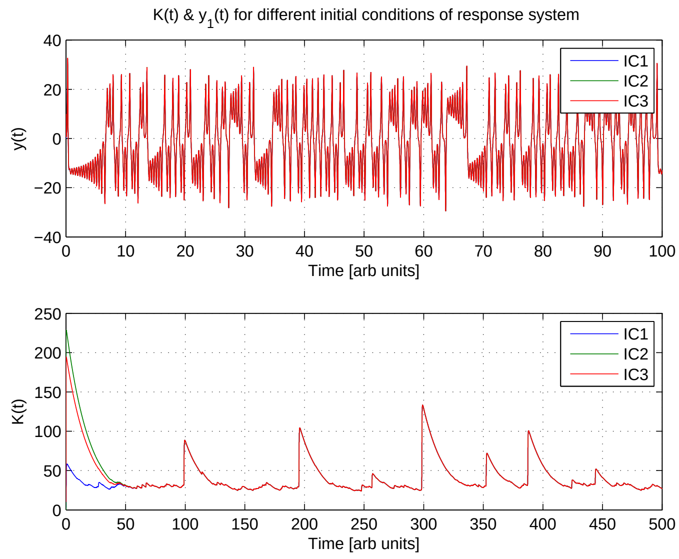


Figure 3.4: Generalized synchronization in the Dynamical Coupling method

The Lorenz "data" system of equations is shown above with labels $x_i$, the $y_i$ and $K(t)$ equations constitute the N+1 dimensional "model" system that will be used for parameter searching. Note that the coupling is only in the $i = 1$ term. The dynamics of the coupling is designed such that $K(t)$ tends toward zero exponentially except when the data and model do not match, "match" being defined by an error less than a sufficiently small quantity given by the symbol $\gamma$. For the purposes here $\gamma$ is set to be 0.05 (about $\frac{1}{1000}$ the size of the attractor). A bound on the driving term of $K(t)$ is modeled using a hyperbolic tangent. Lastly, a coefficient in front of the tanh is chosen to allow $K(t)$ to span a given magnitude. For this example, $\sigma_1$=$\sigma_2$=16, $b_1$=$b_2$=4, $R_1$=45.92 and $R_2$ is used as the scanning parameter. The damping constant, $\alpha$, is empirically set at 0.05 but will be shown to have a range of acceptable values spanning about a decade. A segment of time series data is shown in Figure 3.5 for the case of identical systems ($R_2$=45.92). On this scale there is no significant difference in the data, $x_1(t)$, and the model $y_1(t)$. The plot of the squared error shows sporadic unsynchronized activity between regions of synchronization. The $K(t)$ plot shows how the coupling reacts to these error spikes - growing until the system is synchronized and then decaying back toward zero as designed.

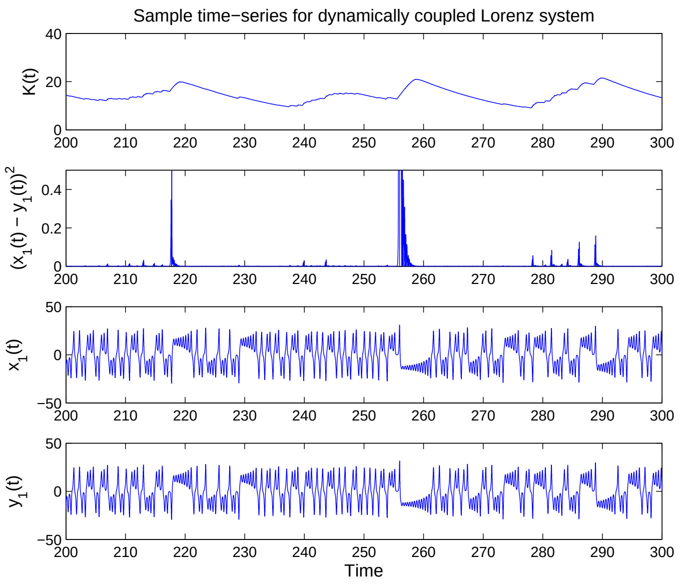


Figure 3.5: Dynamically Coupled Lorenz Time–Series

### 3.3.3 Cost Function Analysis

Given the idea of coupling the measured data into an augmented model–coupling system, it is necessary to consider the cost function that will be most effective for the evaluation of parameters. The error $(x_1 - y_1)$ at any particular point in time may be on the order of the entire size of the attractor, the magnitude of any particular spike dependent on the value of the largest CLE evaluated at the point and time in the orbit where the divergence occurred. Since these local CLE's are not known, it is not necessarily reasonable to assume that a large error spike indicates a large discrepancy in parameters. Therefore, it may not be advantageous to use a cost function like $\int_0^T (x_1 - y_1)^2 dt$ which assumes a specific relationship between phase space error and parameter correctness.

Figure 3.6 shows an expanded section of the squared error vs time over several values of $\mathrm{R}_2$. In addition, a bar at the level of $\gamma^2$ is drawn. The bar is green when the error magnitude is below $\gamma$ and red when above. One possible cost

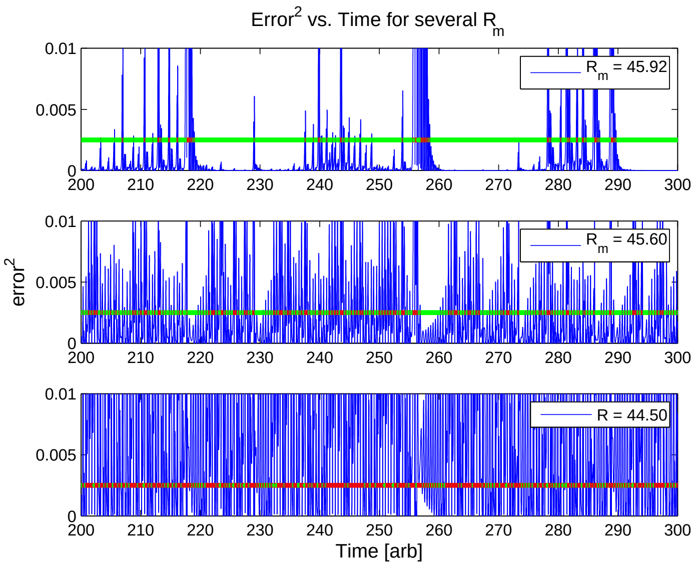


Figure 3.6: Description of cost function for Dynamically Coupled method

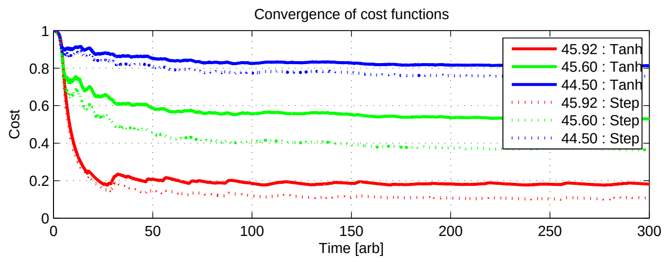


Figure 3.7: Cost function convergence

measure is, over a given time interval, how frequently is the error above $\gamma$? Note how for the correct parameter value of $R_2$=45.92 the bar is mostly green. As this parameter is changed the bar becomes more dominantly red. Using this idea, the cost function could be the ratio of the red segments to the entire interval, or $\frac{1}{T}\int_0^T H(|error| - \gamma)dt$ ($H(\cdot)$ is the Heavyside step function). This cost function is shown in Figure 3.7 in dotted lines. The graph verifies that this cost measure converges quickly after some initial fluctuation. In some sense it is a measure of the probability of looking at the system and observing it to be unsynchronized (a likelihood that we would like to be a minimum). Another option is to use a hyperbolic tangent instead of a discontinuous step function. In this case the cost function could be $\frac{1}{T}\int_0^T \tanh\left[(\frac{(x_1-y_1}{\gamma})^2\right] dt$ which "turns on" when the error is about $\gamma$. The convergence of this cost measure to a steady value is shown in Figure 3.7 in solid lines. Note that both of these cost functions disregard the absolute magnitude of the error and only take into account the magnitude relative to $\gamma$. This new form of cost function is essentially a low–pass filter and allows for a cost function to be evaluated in real–time. One interesting application of this method is discussed in Chapter 4.

With this type of probabilistic cost function defined we may now return to the issue of choosing the damping parameter $\alpha$ of the coupling dynamics. Figure 3.8 shows plots of the average coupling and cost function minimum of identical data and model systems as the damping constant $\alpha$ is varied. In the range $0.01 \leq \alpha \leq 0.1$ the cost function takes on the anticipated average value of $< K(t) > \approx 14$ where the largest CLE becomes just negative (see Figure 2.10). There is a general trend for the cost function minimum to decrease with decreasing $\alpha$ during this interval. For $\alpha > 0.1$ the damping is too strong and $K(t)$ is never able to grow to the value needed for synchronization. Without this generalized synchronization the cost function approaches its maximum value of one, indicating that it is extremely likely that the model system and the data producing system are not synchronized. For $\alpha < 0.01$ the coupling dynamics is too weakly damped and $K(t)$ is allowed to grow very large. With the strong coupling and in the absence of noise, the model and data will eventually become identical to within the precision of the numerical floating point representation. This causes an erroneous result with the average cost function very small and the average coupling large. Therefore, for this example, $\alpha$ is required to be on the interval $0.01 \leq \alpha \leq 0.1$ away from both of these regions.

### 3.3.4 Parameter Scans

With the augmented model–coupling dynamics defined and the hyperbolic tangent form of the cost function selected, we may now proceed to look at this cost function in parameter space. Figure 3.9 shows this cost function as the parameter $R_2$ is varied over the range $30 \to 50$. The response to three different data sets with

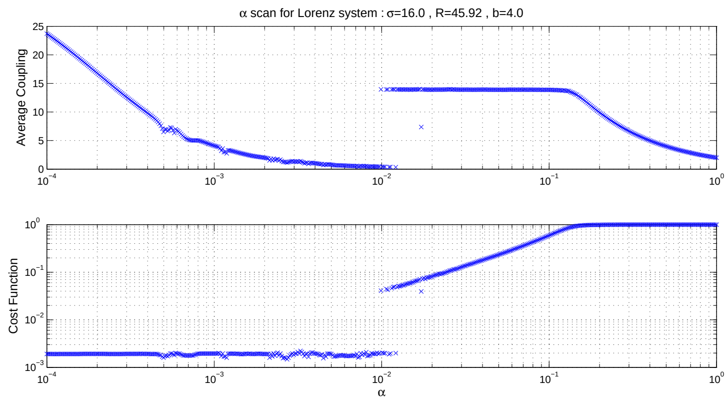


Figure 3.8: Selection of coupling dynamics damping

$R_1 = 34$, $R_1 = 40$ and $R_1 = 45.92$ is shown. In each case there is a very well defined minimum at the location $R_2 = R_1$, increasing steeply toward one on both sides. Figure 3.10 shows a 2D parameter scan of the Lorenz system using a hyperbolic tangent form of the cost function. The data system has parameters $R_1 = 45.0$ and $b_1 = 4.0$ and the model system is scanned over all values $42 \leq R \leq 48$ and $3 \leq b \leq 5$. This plot demonstrates a very well defined minimum at the location $R_2 = R_1$ and $b_2 = b_1$. Figure 3.11 shows parameter scans for Lorenz data with added gaussian measurement noise. Both the hyperbolic tangent form of the cost function and the usual squared error cost function are used. When noise is added, the probabilistic cost function remains smooth with the minimum increasing as the noise level increases. The noise is a source of model–system discrepancy allowing the complex dynamics to cause additional spikes in the error magnitude. These additional spikes result in an increase in the cost function, or an increase in the probability that the model and system are not synchronized. There is a very well defined minimum in the case of the probabilistic cost function throughout all noise levels at the correct value $R_1 = R_2 = 50$, where the squared error cost exhibits many local minima for all noise levels and only a poorly defined minimum around $R_1 = R_2 = 50$.

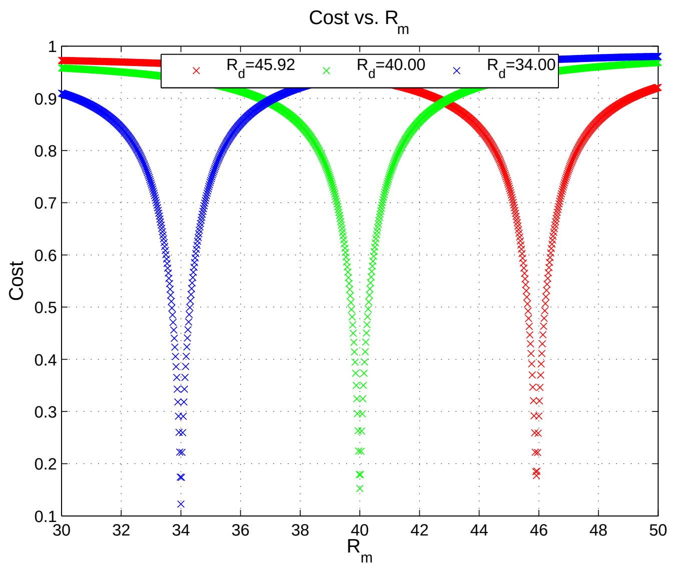


Figure 3.9: Lorenz parameter scans : R

## 3.4 Electrical Circuit Examples

$$\begin{aligned}\dot{\mathrm{x}}_1 &= 10^3[\sigma_1(\mathrm{x}_2-\mathrm{x}_1)] \\ \dot{\mathrm{x}}_2 &= 10^3[\mathrm{R}_1\mathrm{x}_1-\mathrm{x}_2-20\mathrm{x}_1\mathrm{x}_3] \\ \dot{\mathrm{x}}_3 &= 10^3[20\mathrm{x}_1\mathrm{x}_2-\mathrm{b}_1\mathrm{x}_3]\end{aligned}$$

$$\begin{aligned}\dot{\mathrm{y}}_1 &= 10^3[\sigma_2(\mathrm{y}_2-\mathrm{y}_1)+K(t)(\mathrm{x}_1-\mathrm{y}_1)] \\ \dot{\mathrm{y}}_2 &= 10^3[\mathrm{R}_2\mathrm{y}_1-\mathrm{y}_2-20\mathrm{y}_1\mathrm{y}_3] \\ \dot{\mathrm{y}}_3 &= 10^3[20\mathrm{y}_1\mathrm{y}_2-\mathrm{b}_2\mathrm{y}_3] \\ \dot{K} &= -0.05K(t)+4\tanh\left[(\frac{x_1-y_1}{0.1})^2\right]\end{aligned}$$

An electrical circuit implementing the Lorenz system for the data signal (shown above as $\dot{x}_i$) was built to explore dynamical coupling in a real, noisy setting. The factors of 20 and $10^3$ are voltage and time scalings respectively. Here, $\sigma_D = 16$, $\mathrm{R}_D = 50$, and $\mathrm{b}_D = 4$ are the designed parameter values. An NI-DAQ board collected $x_1(t)$ data at a rate of 250KHz for 0.04 seconds. This data is coupled to a numerical model system (shown as $\dot{y}_i$ and $\dot{K}$) and integrated forward in time with arbitrary initial conditions. Here the cost function

$$\text{Cost} = \frac{1}{T}\int_0^T H(|error|-\gamma)dt$$

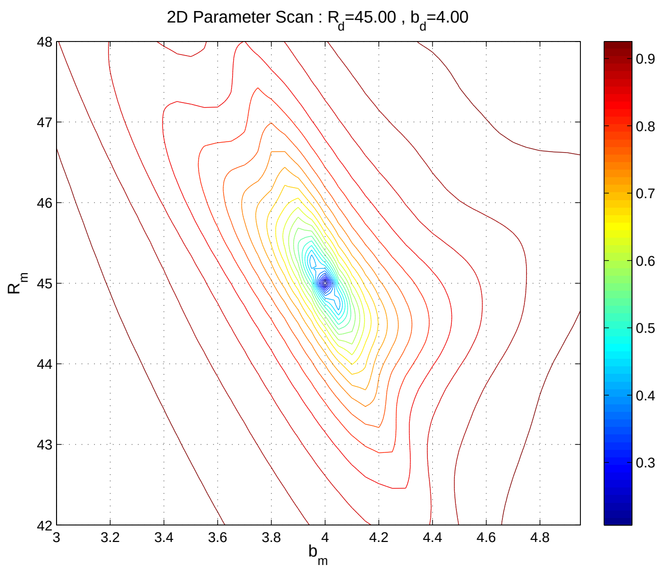


Figure 3.10: Lorenz 2D parameter scans : (R,b)

was used. When $|error| > \gamma$ the systems are considered out of synchronization and the integrand is one, zero otherwise. In this way, the cost function is a measure of the probability of the model being out of synchronization with the system at any point in time. Here $\gamma = 0.05$. Plots showing convergence of this cost function both near and far from its minimum value are shown in Figure 3.12 and Figure 3.13 respectively. The bottom graphs show this cost function rapidly converging to a steady value. Figure 3.14 shows a scan across the parameter $R_M$ for several different data generating Lorenz circuit systems. The values of the minima are shifted to the right slightly from the correct values of 45, 50, and 55. This systematic offset is likely due to the inaccuracies associated with the many components in this circuit. The components used were "off-the-shelf" and taken at face value. Figure 3.15 shows a two dimensional scan of the cost function in the parameter space (R,b). This plot contains a well defined minimum near the correct values of R = 50 and b = 4.0. It must be emphasized that the 'correct values' are actually only predicted values. Inaccuracies in the designed parameters due to variability in electrical component values lead to uncertainty in the actual 'correct values' of parameters.

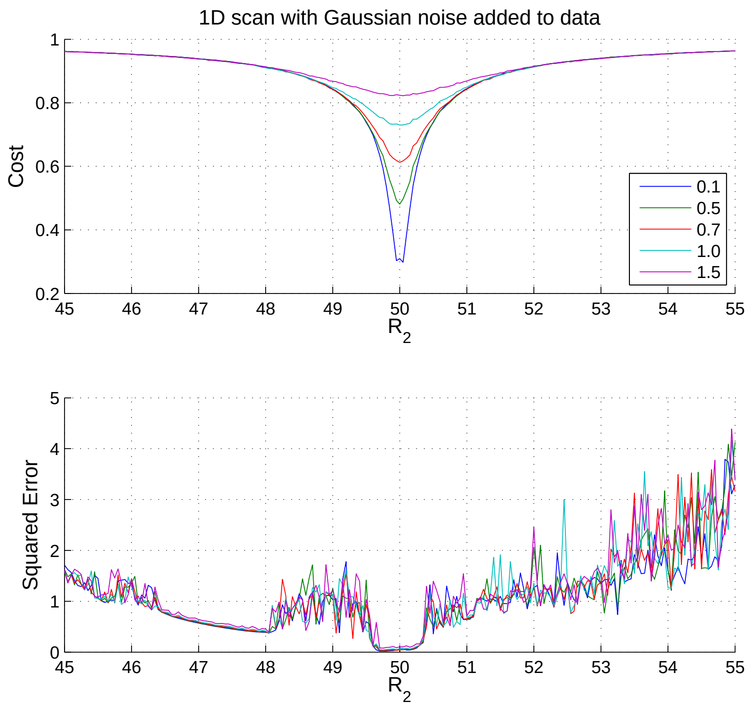


Figure 3.11: Lorenz parameter scans with noise: R

Figure 3.16 demonstrates another example of the dynamical coupling method using data from an electronic circuit. A Colpitts circuit (Figure 2.3) was constructed with component values $C_1 = 6.8\mu F$, $C_2 = 6.8\mu F$, $L = 10mH$, $R_L = 26\Omega$ and with $R_E$ as the unknown parameter. The parameter $R_E$ was set to two values where the system was chaotic. The emitter voltage was sampled at 100,000 kHz and the data was coupled to a numerical model system through the $V_{C2}$ variable using the dynamical coupling method. The graph on the left shows the cost while varying the value of $R_E$ in the numerical system. The correct values of the emitter resistance are $476\Omega$ and $846\Omega$. The offsets are due to inaccuracies in the other component values (especially the capacitors).

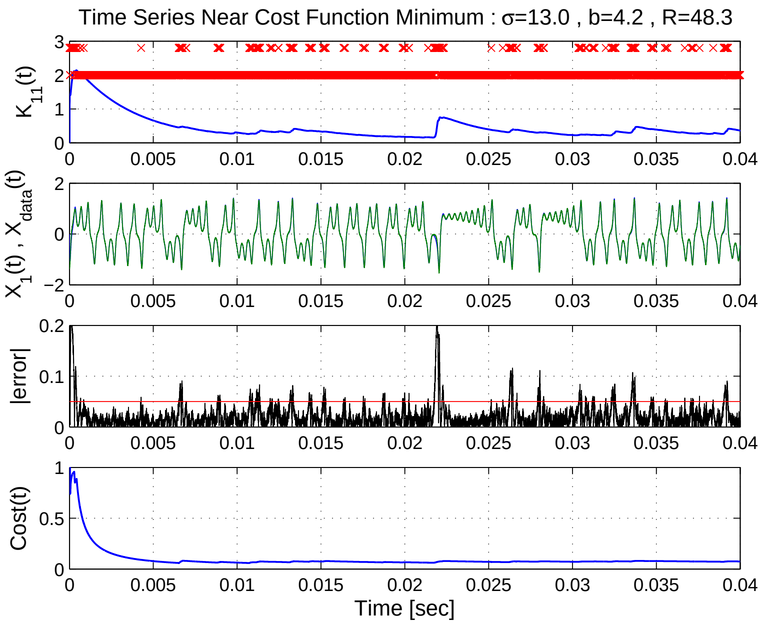


Figure 3.12: Cost function convergence #1 : Lorenz circuit data

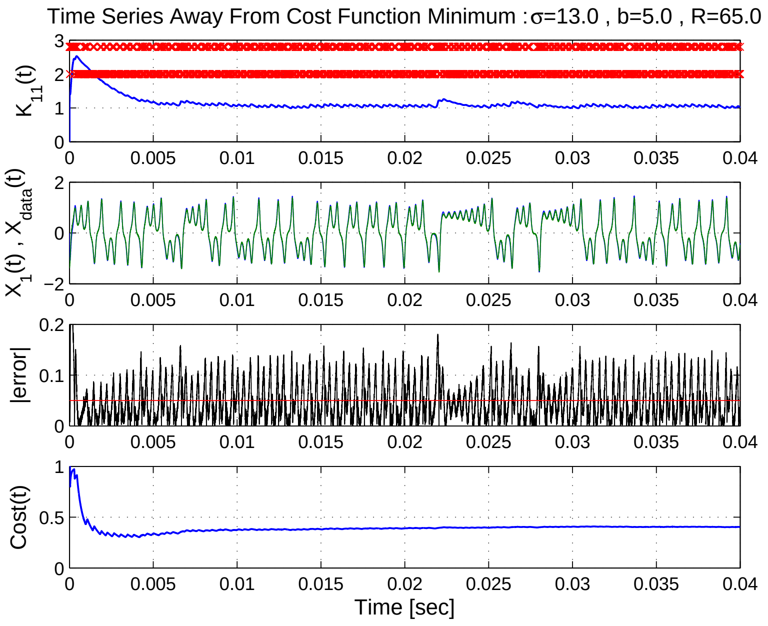


Figure 3.13: Cost function convergence #2 : Lorenz circuit data

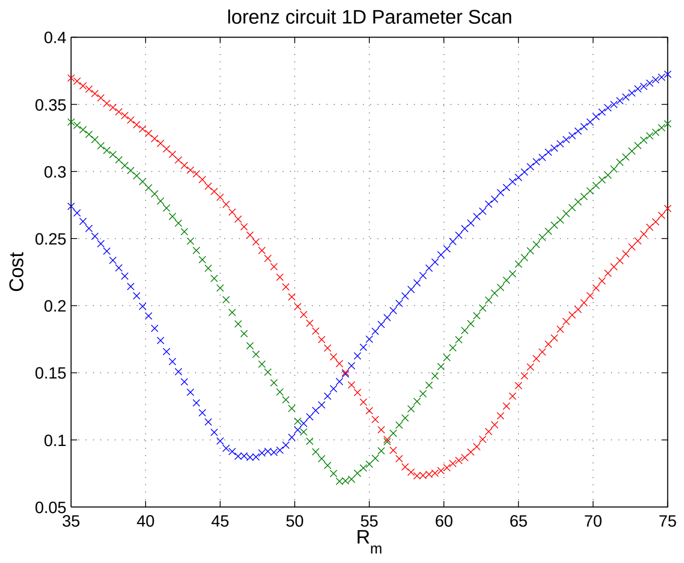


Figure 3.14: Lorenz circuit parameter scan : R

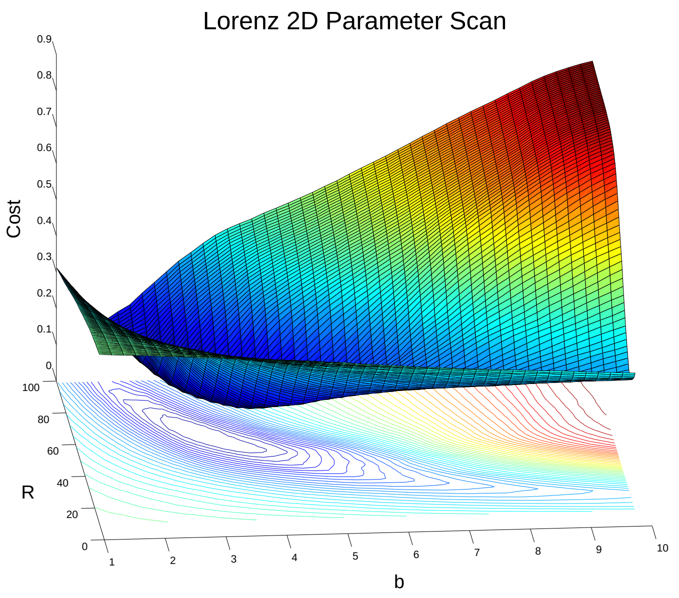


Figure 3.15: Lorenz circuit 2D parameter scan : (R,b)

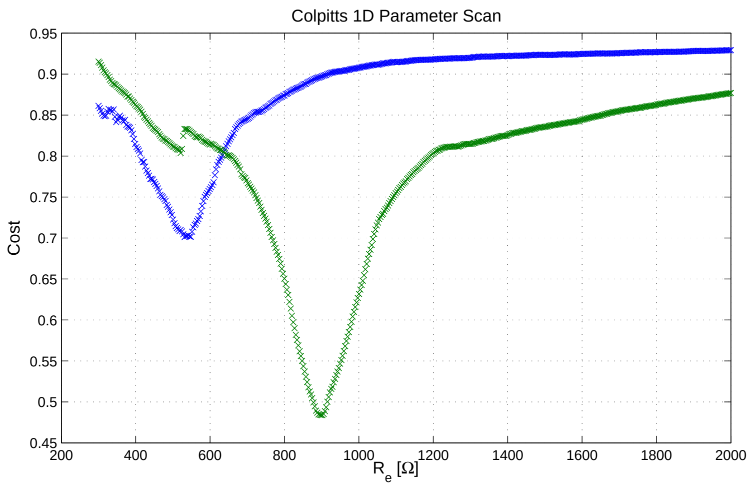


Figure 3.16: Colpitts circuit parameter scan : $R_E$

# Chapter 4

# Application of Dynamical Coupling

## 4.1 Introduction

This chapter focuses on a particular type of engineering application for which the dynamical coupling method is well suited. In this application we wish to quickly distinguish the source of a measured time–series from many candidate sources of identical design. A particular example could be an aircraft radio forced into a nonlinear regime of operation by an intense pulse of electromagnetic energy. In this case, the goal is to distinguish aircraft from one another by the slight differences in the dynamics of their radio systems.

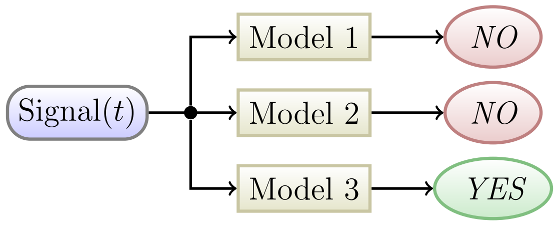


Figure 4.1: System Identification Problem

Here, standing in for the “radio”, the Lorenz system will be used as the source of a complex time–series measurement. Figure 4.1 illustrates the problem. Given a time–series of measured data, distinguish between several candidate models of identical design but slightly different parameters for the model most likely to have generated the measured time–series. A further requirement is that this should be done in real–time. We would like to identify the aircraft in seconds or tens of seconds rather than tens of minutes.

## 4.2 System Identification

A data file representing the transmitting system was generated from the Lorenz system with $4^{th}$ order Runge–Kutta integration and a time step of $\delta = 0.01$. The parameter $\sigma_d$ was set to 16.0 while $\mathrm{R_d}$ and $\mathrm{b_d}$ changed in time as shown in Table 4.1.

$$\begin{aligned}
\dot{x}_1 &= 16(x_2 - x_1) \\
\dot{x}_2 &= \mathrm{R_d} x_1 - x_2 - x_1 x_3 \\
\dot{x}_3 &= x_1 x_2 - \mathrm{b_d} x_3
\end{aligned}$$

Table 4.1: Lorenz system parameters : Simulated data

| Time | $\mathrm{R_d}$ | $\mathrm{b_d}$ |
|---:|---:|---:|
| $0 \rightarrow 300$ | 60 | 4.6 |
| $300 \rightarrow 600$ | 50 | 4.0 |
| $600 \rightarrow 900$ | 52 | 4.1 |
| $900 \rightarrow 1200$ | 54 | 4.2 |
| $1200 \rightarrow 1500$ | 60 | 4.6 |

The time series $x_1(t)$ was then simultaneously presented to three receiver models of the following form

$$\begin{aligned}
\dot{y}_1 &= 16(y_2 - y_1) + \mathrm{K}(x_1 - y_1) \\
\dot{y}_2 &= \mathrm{R_m} y_1 - y_2 - y_1 y_3 \\
\dot{y}_3 &= y_1 y_2 - \mathrm{b_m} y_3
\end{aligned}$$

$$\dot{\mathrm{K}} = -\alpha \mathrm{K} + (\frac{x_1 - y_1}{\gamma})^2$$

With $\alpha = 0.1$, $\gamma = 0.2$ and each model having its own $\mathrm{R_m}$ and $\mathrm{b_m}$ as shown in Table 4.2.

Table 4.2: Receiver model parameters for simulated data

| | $\mathrm{R_m}$ | $\mathrm{b_m}$ |
|---|---|---|
| Model #1 | 50 | 4.0 |
| Model #2 | 52 | 4.1 |
| Model #3 | 54 | 4.2 |

The cost function for each model, which is interpreted as the probability that the model is **not** likely to be the one which generated the data, is the frequency over the past 100 points in time that the error has exceeded a specific value $|error| \geq \gamma$.

$$\mathrm{Cost}(t) = \frac{1}{100} \int_{t-100}^{t} \left\{ \begin{array}{ll} 1 & |x_1 - y_1| \geq \gamma \\ 0 & |x_1 - y_1| < \gamma \end{array} \right\} dt$$

In other words, the cost function is the probability that a randomly chosen point on the interval will have $|error| \geq \gamma$. We desire for this probability to be a minimum indicating that the model is easily driven into synchronization with the data. The width of the averaging window, 100, was chosen to be as small as possible while still allowing the probability calculation to converge to a steady value. For calculation purposes, the cost function was initialized to 1 for $t \leq 0$.

Figure 4.2 shows two example error time–series plots from model #1 where the cost function is evaluated for $t = 500$ and $t = 300$. A horizontal line is drawn at the value $|error| = \gamma$ and is colored red when $|error| > \gamma$ and green when $|error| \leq \gamma$. The cost function is seen as the ratio $\frac{\text{length of red}}{\text{total length}}$. Figure 4.3 shows the time-series responses for $|error(t)|$, coupling $\mathrm{K}(t)$, and Probability$(t)$ (or cost function) for model system #2. Note the minimum in the probability when the data system has switched to parameters equivalent to the model. Also note the spiking activity that takes place in $\mathrm{K}(t)$ when the model is introduced to a favorable time–series. The best cost function, which may not be the particular one

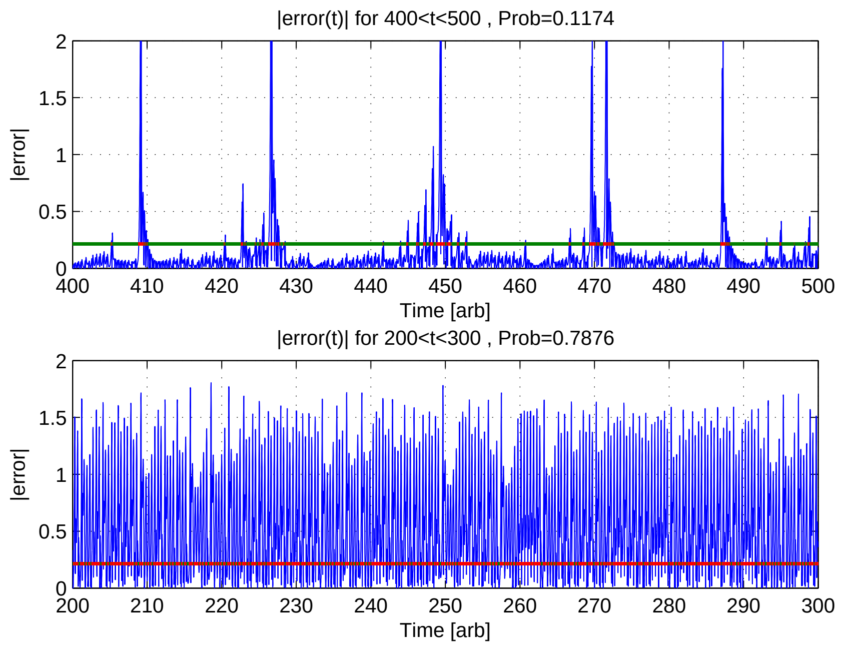


Figure 4.2: Probability cost function

used here, should be able to detect the onset of this behavior in the least amount of time. Figure 4.4 shows the response to each model as time progresses. Note the ability of each model to respond to the portion of the time–series most likely to have been generated by itself.

This system identification strategy uses the dynamical coupling method of introducing measured data to a model as a real–time discriminator. The cost function, being a low–pass filter, may be integrated in parallel with the model systems providing cost function readings on a continuous basis. The measured time–series is continually introduced to the known system models and the models "respond" with a minimum cost function when the measured data is likely to have been produced by that model. As another application, this threshold based discrimination may also be used as part of a communication system to send symbols hidden in a chaotic time–series. Presently the baud rate for such a communication system would be intolerably low due to the time lag in the cost function low–pass filter. Further research may provide an alternative, more responsive, cost function and make the communication application a viable concept.

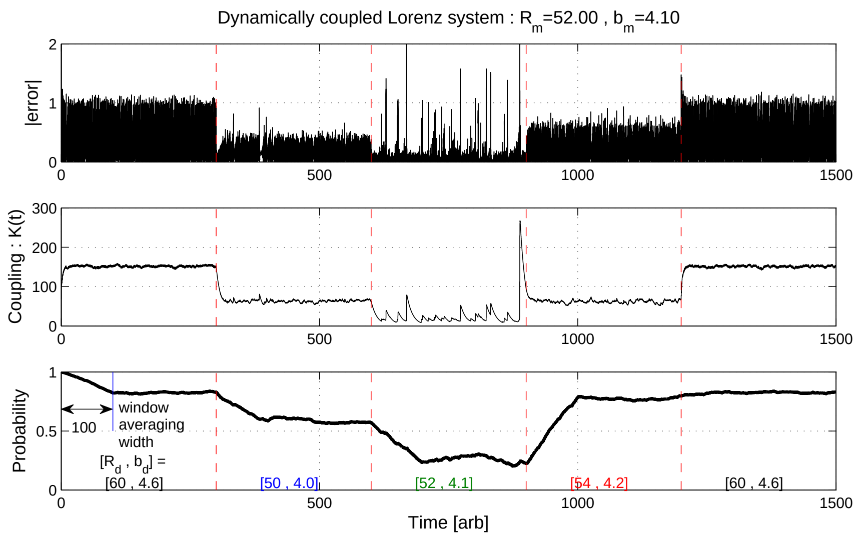


Figure 4.3: Time-series for model #2 (simulated data)

This system identification method was tested with measured data from a Lorenz electrical circuit. Here, the parameters of the data producing system were changed in time by use of a PIC microcontroller programmed to power a series of electronic relays in a predetermined sequence. These relays switched resistors responsible for the magnitude of the parameters R$_d$ and b$_d$ in and out of the circuit. A photograph of the setup is shown in Figure 4.5. The portion of the circuit circled on the left consists of the micro–controller and relays, the portion circled on the right is the actual Lorenz system circuit. Again, Figure 4.6 shows the time-series responses for $|error(t)|$, coupling K$(t)$, and Probability$(t)$. Figure 4.7 shows the response of each numerical model (Table 4.3) to the time–series and agrees with the above simulated results very well.

## 4.3 Parameter Space Contours

The above method of signal discrimination introduces the issue of determining exactly where to "draw the line" at some cost function level to indicate whether

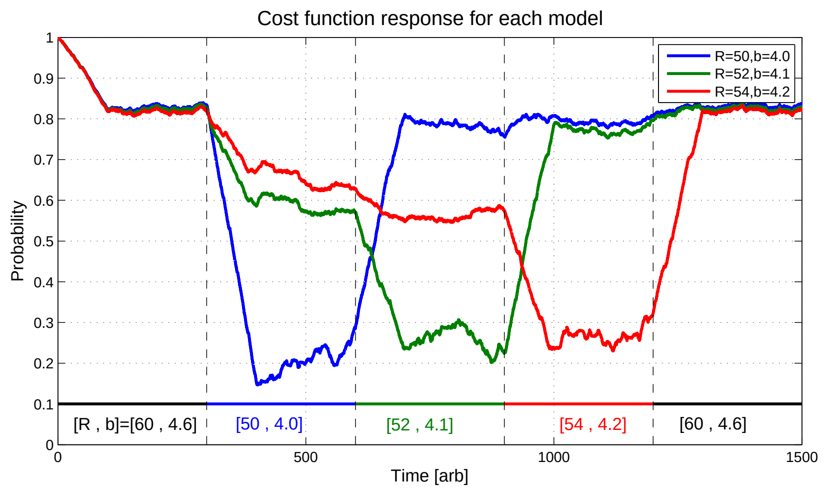


Figure 4.4: Probability time-series for all three models (simulated data)

Table 4.3: Receiver model parameters for circuit data

| | $R_m$ | $b_m$ |
|---|---|---|
| Model #1 | 45.7 | 4.3 |
| Model #2 | 51.6 | 5.1 |
| Model #3 | 57.7 | 5.5 |

a model is responding favorably to a data signal. This critical cost function value would also need to be dependent on the level of background noise in the system at any particular time (Figure 3.11). A better method is to use an entire contour of points in parameter space rather than only a single point. Figure 4.8 depicts the two–dimensional parameter scan in $(R, b)$ for the Lorenz system. We wish to concentrate on a specific contour in parameter space, say the values of $(R, b)$ for which the cost function equals 1/2. The idea is to adjust the parameters of the model system such that it follows a contour of constant cost function when introduced to a favorable time–series measurement. A time–series from a system with slightly different parameters will cause the minimum shown in Figure 4.8 to be shifted. This will result in the cost increasing on one side of the contour and decreasing on the other. This modulation of the cost function value as a result of a changed parameter space landscape is easily identified in the data due to it having

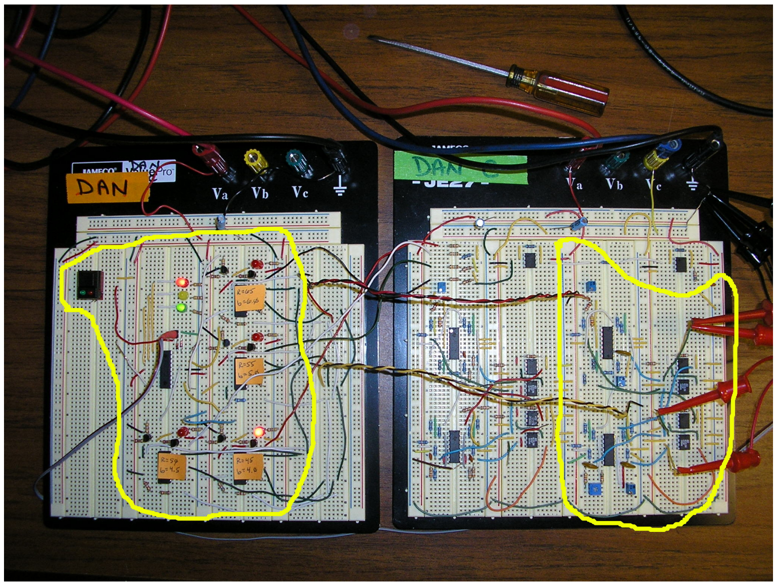


Figure 4.5: Lorenz circuit with micro–controller based parameter adjustment

a known period; the time it takes for the receiver system to progress through the parameter space contour.

Applying a coordinate system with origin inside the contour [36], the points along the contour may be written in terms of the vector $\vec{r} = \rho(\theta)\cos(\theta)\hat{\mathrm{b}} + \rho(\theta)\sin(\theta)\hat{\mathrm{R}}$. Moving along the contour at constant speed requires that the velocity vector satisfy the relationship

$$\left|\frac{d\vec{r}}{dt}\right|^2 = (\rho^2 + \rho'^2)\dot{\theta}^2 = \mathrm{V}_o{}^2$$

Solving this equation for $\dot{\theta}$ and integrating with respect to time gives

$$\theta(t) = \int_0^t \frac{\mathrm{V}_o}{\sqrt{\rho^2 + \rho'^2}} dt$$

The functions $\rho(\theta)$ and $\rho'(\theta)$ are known from the data and we may now numerically integrate to obtain points separated at equal intervals of time moving at constant speed along the parameter contour. Figure 4.9 shows these points on a polar plot where $\theta = [0, 2\pi]$ represents one orbit of the contour. On the right, this same contour is used with data from a system with different parameters. The shifting

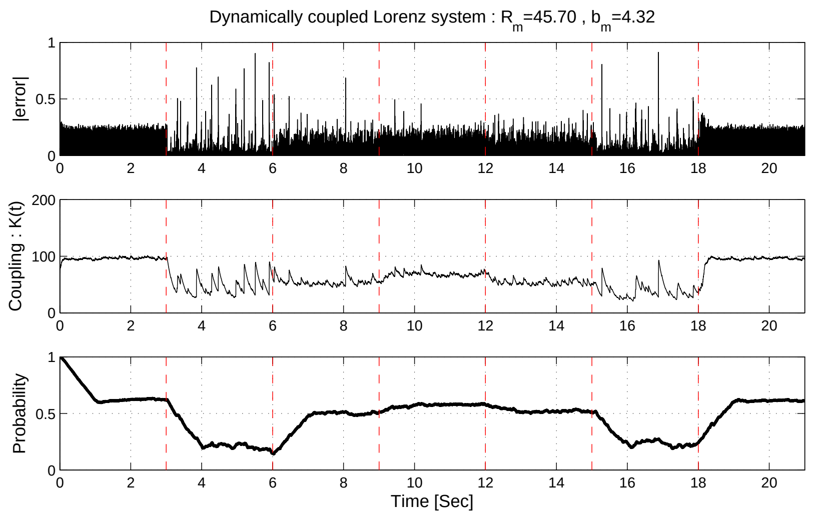


Figure 4.6: Time-series for model #1 (circuit data)

of the cost function minimum is realized as an excursion of the cost function away from the origin on one side of the contour and toward the origin on the other.

As a final example of the sensitivity of this approach, time–series data from seven slightly different Lorenz systems were introduced to a model following a parameter contour calculated from data of the first system. The parameters of each system are presented in Table 4.4. As can be seen in Figure 4.10, the

Table 4.4: Receiver model parameters for contour method

| Parameter | #1 | #2 | #3 | #4 | #5 | #6 | #7 |
|---|---|---|---|---|---|---|---|
| $\sigma$ | 16.00 | 16.00 | 15.72 | 15.84 | 15.84 | 15.96 | 15.98 |
| R | 45.00 | 45.33 | 45.62 | 45.40 | 45.02 | 45.43 | 44.52 |
| b | 4.00 | 4.08 | 3.90 | 4.10 | 4.02 | 3.97 | 3.96 |

correct model system is readily identifiable. Here the cost function is plotted verses normalized length along the contour. The periodic features from incorrect parameters are easily distinguished from the correct model parameters and there is no need to set an arbitrary threshold on the cost function value.

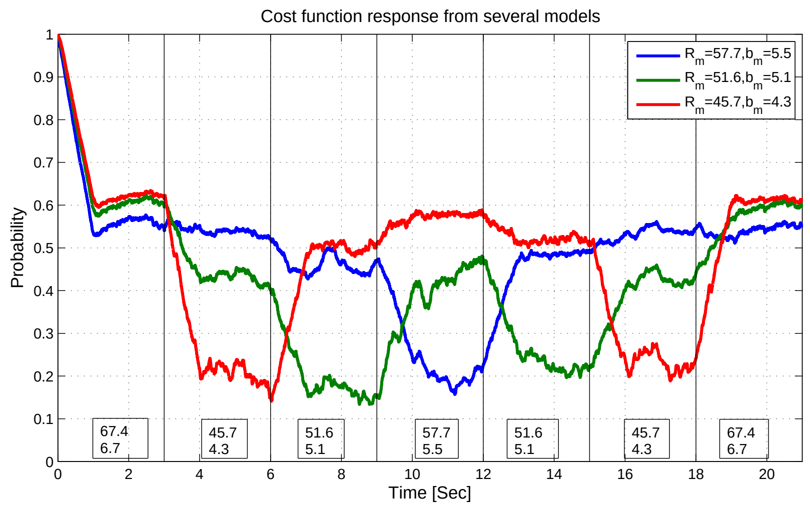


Figure 4.7: Probability time-series for all three models (circuit data)

## 4.4 Summary

This chapter introduced an application of the technique of dynamically coupling measured data into a model system. This method allows for a cost function that may be calculated in parallel with the integration of the dynamics allowing for an immediate prediction (the low–pass filter in the cost function introduces some time lag) of the likelihood that a measured signal could be generated by a candidate model. This is an important result with engineering applications in system identification problems where systems with complex dynamics need to be classified quickly.

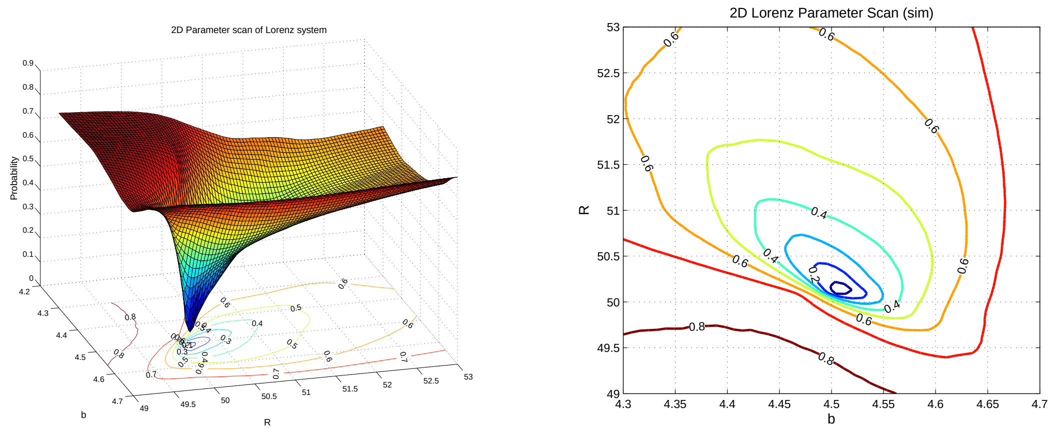


Figure 4.8: 2D Parameter scan with contours

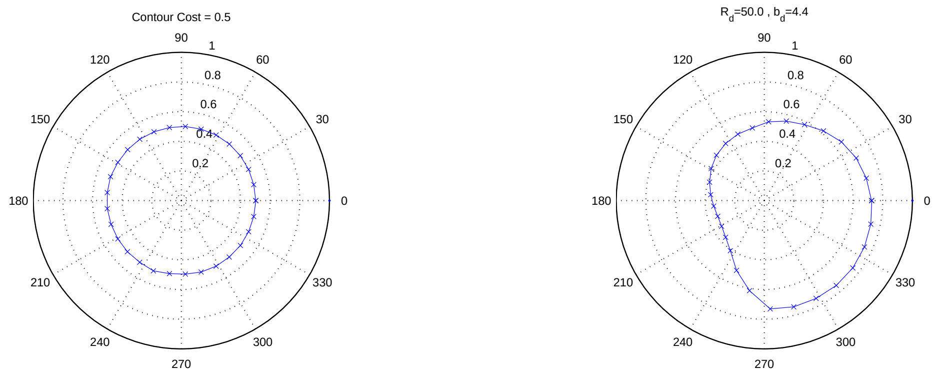


Figure 4.9: Polar plots of original and distorted parameter contours

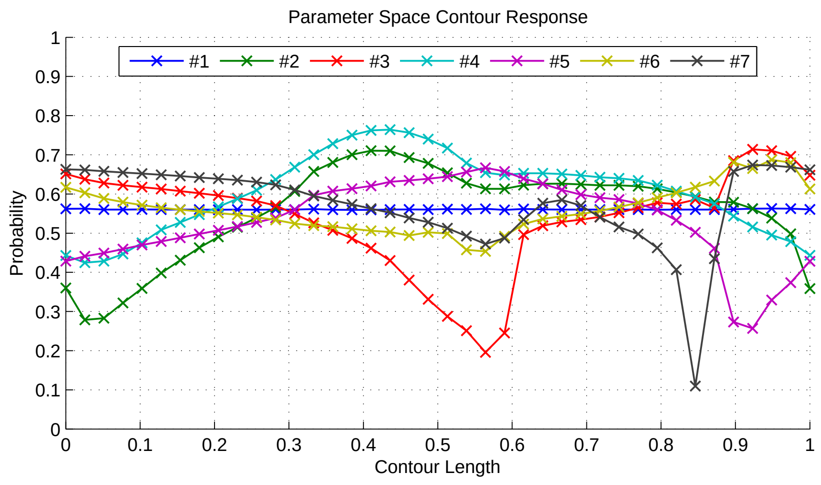


Figure 4.10: Parameter space contour system identification : Simulated results

# Chapter 5

# Optimal Tracking Formulation

This chapter introduces a technique for solving the state and parameter estimation problem via an analogy with classical optimal control theory [17, 20, 6]. The model system is thought of as being controlled in such a way as to force it to track measured data. Once the model system is discretized in the appropriate way and the correct cost function chosen, powerful existing numerical optimization code may be used to solve for the optimal control that allows for the model to reproduce the measured data time–series. This control function may then be used as a means of comparing several candidate system models. Examples are provided for the Colpitts, Lorenz, and Hodgkin–Huxley systems.

## 5.1 Optimal Tracking of Data

Chapter 3 introduced a constant term $K$ to couple measured data to a model system. This idea was then extended to a time–dependent coupling gain $K(t)$ but with constrained time evolution through an ad–hoc differential equation for $\dot{K}(t)$. Here the coupling gain is allowed to be an arbitrary positive valued function of time $u(t)$. Note that the symbol $u$ has replaced $K$ as representing the proportional error coupling gain to emphasize that it is now an unconstrained function of time.

Classical optimal control problems are typically cast in the following form. There is some input/output system we wish to control, called the "plant". There is the "controller" which provides input to the plant based on current measurements

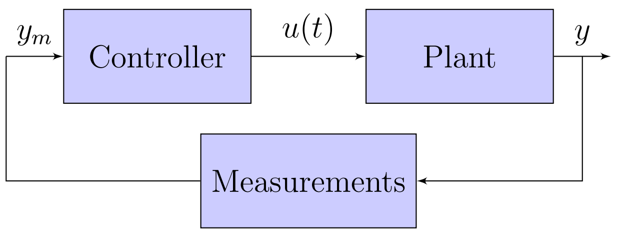


Figure 5.1: Optimal control system

of the output and possibly the time. A typical diagram is shown in Figure 5.1. The problem is to design a controller to bring the plant from some initial state to a particular final state while minimizing some performance measure (5.1).

$$\mathbf{J} = h(\mathbf{x}(t_f), t_f) + \int_{t_o}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \tag{5.1}$$

The performance measure takes into consideration the specific optimization required of the motion. For example, using the least amount of fuel or traveling the least distance. These problems tend to be numerical in nature and optimization tools have been developed for use in solving optimal control problems. Casting the state and parameter estimation problem in an optimal control setting allows the use these powerful numerical tools. If we allow the "plant" to be the system model and the "control" to be the proportional error coupling gain we can then ask for the control which drives the model to track the data in some optimal way.

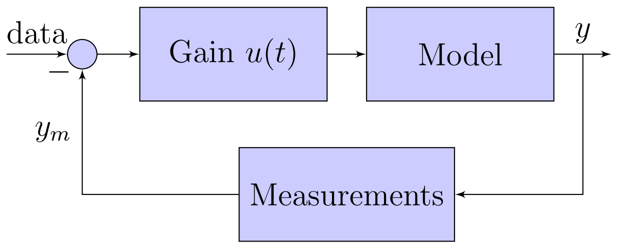


Figure 5.2: Control diagram for model system

Figure 5.2 depicts the control diagram for introducing data to the model. Ideally, we would like the model to track the data on its own without the need for coupling ($u(t) = 0$). This ideal case is very unlikely considering measurement errors and allowing for the possibility of chaotic dynamics in the system under study. For the case of state and parameter estimation, the optimal control will be the $u(t)$ which allows the model system to track the measured data while also being as small as possible. We only require the control to facilitate tracking noisy data and chaotic dynamics, we do not want the control to dominate the dynamics due to the inability of the model to reproduce the data. Indeed, the resulting $u(t)$ required to force the model to track measured data will be an indication of how "good" the model is, and may serve as a metric for comparing several candidate models with each other.

The observed system is assumed to satisfy differential equations in $\mathbf{x}(t)$ that depend on a fixed set of parameters $\mathbf{p}$, i.e.,

$$\begin{aligned}\frac{dx_1(t)}{dt} &= F_1(x_1(t), \mathbf{x}_\perp(t), \mathbf{p}) \\ \frac{d\mathbf{x}_\perp(t)}{dt} &= \mathbf{F}_\perp(x_1(t), \mathbf{x}_\perp(t), \mathbf{p}).\end{aligned}$$

The measured quantity $x_1(t)$ is passed to the model by augmenting the model dynamics with a proportional error term as shown in equations (5.2).

$$\begin{aligned}\frac{dy_1(t)}{dt} &= F_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}) + u(t)(x_1(t) - y_1(t)) \\ \frac{d\mathbf{y}_\perp(t)}{dt} &= \mathbf{F}_\perp(y_1(t), \mathbf{y}_\perp(t), \mathbf{q})\end{aligned} \tag{5.2}$$

We wish to determine the coupling control, $u(t)$, and model parameters, denoted by $\mathbf{q}$, such that the model system tracks the data in an optimal fashion. As stated above, "optimal" is defined as the smallest $u(t)$ which allows the model to track the measured data. We require a cost function which, in addition to a penalty for large error, also contains a penalty for large control values. One such cost function is shown in equation (5.3).

$$C(\mathbf{q}, u) = \frac{1}{2T}\int_0^T [(x_1(t) - y_1(t))^2 + u(t)^2] \tag{5.3}$$

For large $u(t)$ the first term in this function behaves as $1/u^2$, and this, combined with the growth of the second term, leads to a balanced magnitude for $u(t)$. By minimizing the cost function subject to the constraint of satisfying the differential equations (5.2), we obtain a classical tracking problem with optimal control [17]. The trajectory of the dynamical system for $\mathbf{y}(t)$ is controlled by $u(t)$ to track $x_1(t)$, or, in contemporary language, to synchronize with the observed orbit $x_1(t)$. Because of the properties of nonlinear systems, when the largest CLE of (5.2) is negative, the unobserved components of the state, i.e., $\mathbf{y}_\perp(t)$, will also track the unobserved $\mathbf{x}_\perp(t)$.

## 5.2 System Discretization

To define the optimal control problem we assume measurements are made of the system at times $t_m = t_0 + m\tau$, resulting in $\mathbf{X}(m) = \mathbf{x}(t_0 + m\tau)$; $m = 1, \ldots, M$. These measurements could be an arbitrary scalar function of the system state, but to simplify the discussion we assume here that $x_1(t)$ is observed as the data. The state of the model is described by $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_N(t)] = [y_1(t), \mathbf{y}_\perp(t)]$. Where $U(m) = u(t_0 + m\tau)$ are the control values defined at the discrete measurement times.

To solve the optimization problem, we use a “direct transcription” method [11], which defines a finite-dimensional problem with variables given by the states $\mathbf{Y}(m)$, the controls $U(m)$, and the fixed parameters $\mathbf{q}$. The cost function (5.3) is minimized subject to equality constraints that connect the $\mathbf{Y}(m)$ and the $U(m)$ across each time interval $[t_m, t_{m+1}]$. These constraints are imposed in the finite-dimensional space of the variables $\{\mathbf{Y}(m), U(m), \mathbf{q}\}$ and are characterized by an integration rule for the states $\mathbf{Y}(m)$, and an interpolation rule for the control $u(t)$. Although the resulting finite-dimensional optimization problem has many variables, it is also smooth and has derivatives that may be calculated efficiently. Indeed, it is this transcription into a smooth large-scale constrained optimization problem that is the key to the robust and efficient estimation of the parameters.

This work uses Simpson’s rule for the integration and Hermite interpolation for the states and controls. Given the values of the states and controls at the mid-point $m2 = m + \frac{1}{2}$, Simpson’s rule for integration defines a constraint on each interval of the form:

$$\begin{aligned}\mathbf{Y}(m) + \frac{\tau}{6}[&\mathbf{G}(\mathbf{Y}(m), U(m), \mathbf{q}) \\ &+ \mathbf{G}(\mathbf{Y}(m+1), U(m+1), \mathbf{q}) \\ &+ 4\mathbf{G}(\mathbf{Y}(m2), U(m2), \mathbf{q})] - \mathbf{Y}(m+1) = 0,\end{aligned}$$

where

$$\begin{aligned}\mathbf{G}_1(\mathbf{y}, u, \mathbf{q}) &= \mathbf{F}_1(\mathbf{y}, \mathbf{q}) + u(\mathbf{x} - \mathbf{y}) \\ \mathbf{G}_\perp(\mathbf{y}, u, \mathbf{q}) &= \mathbf{F}_\perp(\mathbf{y}, \mathbf{q}).\end{aligned} \tag{5.4}$$

In order to determine $U(m)$ and the mid–point values, $\mathbf{Y}(m2)$, $\mathbf{Y}(m)$, $U(m2)$, we require a formula for interpolating the model states and control on the interval $t \in [n\tau, (n+1)\tau]$. Using cubic polynomial interpolation we may approximate functions as $y(t) = \mathrm{A}t^3 + \mathrm{B}t^2 + \mathrm{C}t + \mathrm{D}$, with the coefficients determined from the conditions $y(m\tau) = Y(m)$, $y((m+1)\tau) = Y(m+1)$, $\dot{y}(m\tau) = G(m)$ and $\dot{y}((m+1)\tau) = G(m+1)$. A general point along the interval $t = [m\tau, (m+1)\tau]$ denoted by $s \in [0,1]$ is approximated by

$$\begin{aligned}\mathbf{y}(s) \approx\ & [\tau(\mathbf{G}(m) + \mathbf{G}(m+1)) + 2(\mathbf{Y}(m) - \mathbf{Y}(m+1))]\, s^3 \\ & + [-\tau(2\mathbf{G}(m) + \mathbf{G}(m+1)) - 3(\mathbf{Y}(m) - \mathbf{Y}(m+1))]\, s^2 \\ & + [\tau\mathbf{G}(m)]\, s + \mathbf{Y}(m)\end{aligned} \tag{5.5}$$

For approximation of the mid–point value, $s = 1/2$ and the above equation reduces to

$$\mathbf{y}(s = \frac{1}{2}) \approx \frac{1}{2}[\mathbf{Y}(n) + \mathbf{Y}(n+1)] + \frac{\tau}{8}[\mathbf{G}(n) - \mathbf{G}(n+1)] \tag{5.6}$$

$\mathbf{Y}(m2)$, $\mathbf{Y}(m)$, $U(m2)$ and $U(m)$ are required to satisfy this Hermite interpolation condition [35], giving

$$\begin{aligned}\mathbf{Y}(m2) = \ & \frac{1}{2}(\mathbf{Y}(m) + \mathbf{Y}(m+1)) + \frac{\tau}{8}(\mathbf{G}(\mathbf{Y}(m), U(m), \mathbf{q}) \\ & - \mathbf{G}(\mathbf{Y}(m+1), U(m+1), \mathbf{q})),\end{aligned}$$

and

$$U(m2) = \frac{1}{2}(U(m) + U(m+1)) + \frac{1}{8}(\mathbf{dU}(m) - \mathbf{dU}(m+1)),$$

where $du(m)$ is the slope parameter in the Hermite interpolation and is treated as another vector of unknown values.

## 5.3 SNOPT Solver

To illustrate the procedure of setting up and solving the optimization problem let us consider the example of the Lorenz system. The above discretization method produces the following equality constraints that need to be satisfied. From the Simpson integration rule

$$\begin{aligned}
0 &= \mathbf{Y}_1(m) - \mathbf{Y}_1(m+1) + \frac{\tau}{6}\left[\sigma(\mathbf{Y}_2(m) - \mathbf{Y}_1(m)) + U(m)(\mathbf{X}_1(m) - \mathbf{Y}_1(m))\right] \\
&\quad + \frac{2\tau}{3}\left[\sigma(\mathbf{Y}_2(m2) - \mathbf{Y}_1(m2)) + U(m2)(\mathbf{X}_1(m2) - \mathbf{Y}_1(m2))\right] \\
&\quad + \frac{\tau}{6}\left[\sigma(\mathbf{Y}_2(m+1) - \mathbf{Y}_1(m+1)) + U(m+1)(\mathbf{X}_1(m+1) - \mathbf{Y}_1(m+1))\right] \\
0 &= \mathbf{Y}_2(m) - \mathbf{Y}_2(m+1) + \frac{\tau}{6}\left[\mathrm{R}\mathbf{Y}_1(m) - \mathbf{Y}_2(m) - \mathbf{Y}_1(m)\mathbf{Y}_2(m)\right] \\
&\quad + \frac{2\tau}{3}\left[\mathrm{R}\mathbf{Y}_1(m2) - \mathbf{Y}_2(m2) - \mathbf{Y}_1(m2)\mathbf{Y}_2(m2)\right] \\
&\quad + \frac{\tau}{6}\left[\mathrm{R}\mathbf{Y}_1(m+1) - \mathbf{Y}_2(m+1) - \mathbf{Y}_1(m+1)\mathbf{Y}_2(m+1)\right] \\
0 &= \mathbf{Y}_3(m) - \mathbf{Y}_3(m+1) + \frac{\tau}{6}\left[\mathbf{Y}_1(m)\mathbf{Y}_2(m) - \mathrm{b}\mathbf{Y}_3(m)\right] \\
&\quad + \frac{2\tau}{3}\left[\mathbf{Y}_1(m2)\mathbf{Y}_2(m2) - \mathrm{b}\mathbf{Y}_3(m2)\right] \\
&\quad + \frac{\tau}{6}\left[\mathbf{Y}_1(m+1)\mathbf{Y}_2(m+1) - \mathrm{b}\mathbf{Y}_3(m+1)\right]
\end{aligned}$$

and from the Hermite interpolation rule

$$\begin{aligned}
0 &= \frac{1}{2}(\mathbf{Y}_1(m) + \mathbf{Y}_1(m+1)) - \mathbf{Y}_1(m2) \\
&\quad + \frac{\tau}{8}\left[\sigma(\mathbf{Y}_2(m) - \mathbf{Y}_1(m)) + U(m)(\mathbf{X}_1(m) - \mathbf{Y}_1(m))\right] \\
&\quad - \frac{\tau}{8}\left[\sigma(\mathbf{Y}_2(m_1) - \mathbf{Y}_1(m+1)) + U(m+1)(\mathbf{X}_1(m+1) - \mathbf{Y}_1(m+1))\right] \\
0 &= \frac{1}{2}(\mathbf{Y}_2(m) + \mathbf{Y}_2(m+1)) - \mathbf{Y}_2(m2) + \frac{\tau}{8}\left[\mathrm{R}\mathbf{Y}_1(m) - \mathbf{Y}_2(m) - \mathbf{Y}_1(m)\mathbf{Y}_2(m)\right] \\
&= -\frac{\tau}{8}\left[\mathrm{R}\mathbf{Y}_1(m+1) - \mathbf{Y}_2(m+1) - \mathbf{Y}_1(m+1)\mathbf{Y}_2(m+1)\right] \\
0 &= \frac{1}{2}(\mathbf{Y}_3(m) + \mathbf{Y}_3(m+1)) - \mathbf{Y}_3(m2) + \frac{\tau}{8}\left[\mathbf{Y}_1(m)\mathbf{Y}_2(m) - \mathrm{b}\mathbf{Y}_3(m)\right] \\
&= -\frac{\tau}{8}\left[\mathbf{Y}_1(m+1)\mathbf{Y}_2(m+1) - \mathrm{b}\mathbf{Y}_3(m+1)\right] \\
0 &= \frac{1}{2}(\mathbf{U}(m) + \mathbf{U}(m+1)) - \mathbf{U}(m2) + \frac{\tau}{8}\left[\mathbf{dU}(m) - \mathbf{dU}(m+1)\right]
\end{aligned}$$

where $\mathbf{dU}(m)$ is the slope parameter in the Hermite interpolation. To insure smoothness of the control function $U(m)$ satisfying the Hermite interpolation it is helpful to add terms to the cost function involving $du(m)^2$, which we do in the cost function below.

$$\begin{aligned} C &= \frac{1}{2}\sum_{m=1}^{M}[(X_1(m)-Y_1(m))^2+U(m)^2+du(m)^2 \\ &\quad +(X_1(m2)-Y_1(m2))^2+U(m2)^2], \end{aligned}$$

If we have a total of $M+1$ co–location points then there are $5M+5$ unknown values in $[\mathbf{X}_1(m), \mathbf{X}_2(m), \mathbf{X}_3(m), \mathbf{U}(m), \mathbf{dU}(m)]$ and an additional $4M$ unknowns in $[\mathbf{X}_1(m2), \mathbf{X}_2(m2), \mathbf{X}_3(m2), \mathbf{U}(m2)]$. With the P unknown parameters from the model itself we have a total of $9M+5+\mathrm{P}$ unknown quantities. There are $3M$ Simpson integration constraints and $4M$ Hermite interpolation constraints giving a total of $7M$ constraint equations. The number of co–location points is typically of order hundreds, so that the total number of unknown quantities in this example is on the order of thousands. At first this may not seem like gaining any advantage, going from three unknown values to thousands. The key is in the efficiency of the SNOPT solver which takes advantage of the sparseness of the jacobian matrix of the constraints as shown below. All solutions presented below took on the order of minutes to tens of minutes to solve on a standard Linux PC.

| Jac | $\mathbf{x}_1$ | $\mathbf{x}_2$ | $\mathbf{x}_3$ | $\mathbf{u}$ | $\mathbf{du}$ | $\mathbf{x}_{1,2}$ | $\mathbf{x}_{2,2}$ | $\mathbf{x}_{3,2}$ | $\mathbf{u}_2$ | $\mathbf{p}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{C}_{S1}$ | $\mathbf{A}$ | | 0 | | 0 | | | 0 | | |
| $\mathbf{C}_{S2}$ | | | | 0 | 0 | | | | 0 | |
| $\mathbf{C}_{S3}$ | | | | 0 | 0 | | | | 0 | $\mathbf{C}$ |
| $\mathbf{C}_{H1}$ | | | 0 | | | $-\mathbb{I}$ | 0 | 0 | | |
| $\mathbf{C}_{H2}$ | | | $\mathbf{B}$ | 0 | 0 | 0 | $-\mathbb{I}$ | 0 | 0 | |
| $\mathbf{C}_{H3}$ | | | | 0 | 0 | 0 | 0 | $-\mathbb{I}$ | 0 | |
| $\mathbf{C}_{Hu}$ | 0 | 0 | 0 | | | 0 | 0 | 0 | | 0 |
| $C_{obj}$ | | 0 | 0 | | | | 0 | 0 | | 0 |

Figure 5.3: Constraint Jacobian for Lorenz system (Hermite-Simpson)

Figure 5.3 shows the jacobian of the constraint equations for Hermite-Simpson discretization of the Lorenz systems of equations. Each entry represents an $mrM \times$ M matrix, some of which were filled in with their respective zero or identity values. Let us examine a few of the entries that are dependent on the constraints. In

| | $x_1(0)$ | $x_1(1)$ | $x_1(2)$ | $\cdots$ | $x_1(M-1)$ | $x_1(M)$ |
|---|---|---|---|---|---|---|
| $\mathrm{C}_{S1}(0)$ | $\bullet$ | $\bullet$ | 0 | $\cdots$ | 0 | 0 |
| $\mathrm{C}_{S1}(1)$ | 0 | $\bullet$ | $\bullet$ | $\cdots$ | 0 | 0 |
| $\vdots$ | | | | $\ddots$ | | |
| $\mathrm{C}_{S1}(M-1)$ | 0 | 0 | 0 | $\cdots$ | $\bullet$ | $\bullet$ |

Figure 5.4: Structure of jacobian entry ‘A’

region ‘A’ of the constraint jacobian we take the derivative of the first Simpson constraint equation with respect to the first state variable, $\partial \mathrm{C}_{S1}(i)/\partial \mathbf{X}_1(j)$. Since this constraint is integrating the state from $m$ to $m+1$, this portion of the jacobian is only nonzero on a band near the diagonal as shown in Figure 5.4. Similarly for region ‘B’ where we take the derivative of the second Hermite constraint with respect to the third state variable, $\partial \mathrm{C}_{H2}(i)/\partial \mathbf{X}_3(j)$. The only nonzero entries are near the diagonal and the matrix has the same form as that shown in Figure 5.4 for region ‘A’. Figure 5.5 shows an entry from the parameter column where we take $\partial \mathrm{C}_{S3}(i)/\mathbf{q}(j)$. For this Lorenz system example, the only non–zero entries are in the column for the parameter b.

By defining the optimization problem in the space of the discretized system model and using a numerical integration rule along with an interpolation rule as constraints, we have gained the advantage of having a problem with a very sparse constraint jacobian where powerful numerical optimization tools exist such as SNOPT [Philip Gill, UCSD] which exploit this sparse structure. As a practical matter of actually coding these derivatives in computer code, a Python [19, 22] front–end to the SNOPT FORTRAN libraries was created. Python is a freely available scripting language commonly used in the field of numerical computation. The front–end that was developed takes advantage of the object–oriented nature of the programming language for setting up the problem efficiently as well as a symbolic mathematics module for evaluating the derivatives. To define and solve an optimization problem, only a small header section of code need change. An example of this section of the code describing the problem is shown in Figure 5.6 with the entire code listing in Appendix C.

| | R | $\sigma$ | b |
|---|---|---|---|
| $C_{S3}(0)$ | 0 | 0 | ● |
| $C_{S3}(1)$ | 0 | 0 | ● |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $C_{S3}(M-1)$ | 0 | 0 | ● |

Figure 5.5: Structure of jacobian entry 'C'

## 5.4 Simulations

This section presents a gallery of results from using the optimal control method of parameter and state estimation presented above on the Colpitts, Lorenz, and Hodgkin–Huxley systems.

### 5.4.1 Colpitts system

As an example of solving for the states of the Colpitts system, Figure 5.7 gives the state variables $\mathbf{Y}(m)$, the transmitted data $X_1(m)$, and the unknown state variable data $X_2(m)$ and $X_3(m)$. Also shown is the control $U(m)$. It is important that the use of SNOPT allows us to deduce good estimates for the **unobserved** dynamical variables $x_2(t)$ and $x_3(t)$. Here the model equations were purposely integrated with fixed initial conditions in order to demonstrate the ability of SNOPT to track data with a transient coming from an unknown initial condition. The rate at which the trajectory for $\mathbf{y}(t)$ approaches the attractor for the data is seen in Figure 5.7, and this is a result of the control which is operating to enforce synchronization.

### 5.4.2 Lorenz system

Here a model Lorenz system was controlled to track data from a simulated Lorenz system with presumed unknown parameters. In Figure 5.8 the plots of the data system and the SNOPT output are exactly on top of one another and the model parameters were all determined correctly to within a few percent (Table 5.1). Note the very small magnitude of control needed for tracking, order $10^{-3}$.

```
#########LORENZ#HERMITE#SIMPSON###########
#
#   Vector Field
Feqnstr = []
Feqnstr.append("sigma*(x2 - x1)")
Feqnstr.append("R*x1 - x2 - x1*x3")
Feqnstr.append("x1*x2 - b*x3")
#
#   String Symbols
Lvars = ["x1","x2","x3"]
Lparams = ["sigma","R","b"]
#
#   Parameter values [lower, init, upper]
Lparamvals = [[1.00 , 2.0 , 100.0],\
              [1.00 , 2.0 , 100.0],\
              [1.00 , 2.0 , 100.00]]
#
#   Time step
hstep = 0.004
#
#   Data File
fName = "data_file.dat"
#
############################################
```

Figure 5.6: Python system definition header

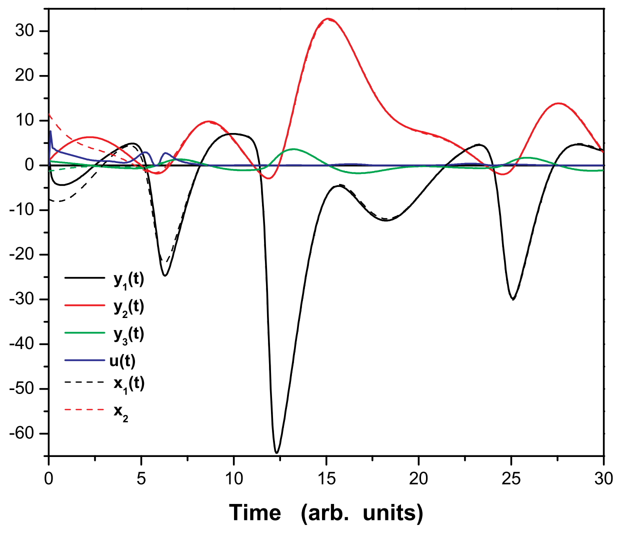


Figure 5.7: SNOPT solution : Colpitts oscillator

Table 5.1: Parameters for Lorenz simulation

| | Data | SNOPT |
|---|---|---|
| $\sigma$ | 15.0 | 15.44 |
| R | 50.0 | 49.95 |
| b | 4.50 | 4.44 |

### 5.4.3 Hodgkin–Huxley model

Here the Hodgkin–Huxley system was controlled to track data from a simulated Hodgkin–Huxley system with unknown parameters and a constant known injection current. The coupling was through the simulated membrane potential $V_m(t)$. In Figure 5.9 the plots of the data system and the SNOPT output are exactly on top of one another. Table 5.2 presents the model parameters involved in this simulation. Several parameters are determined well by SNOPT while others are off by more that 30%.

In this case, one should be concerned with how much information about the system is actually being presented to the model when the neuron is firing period-

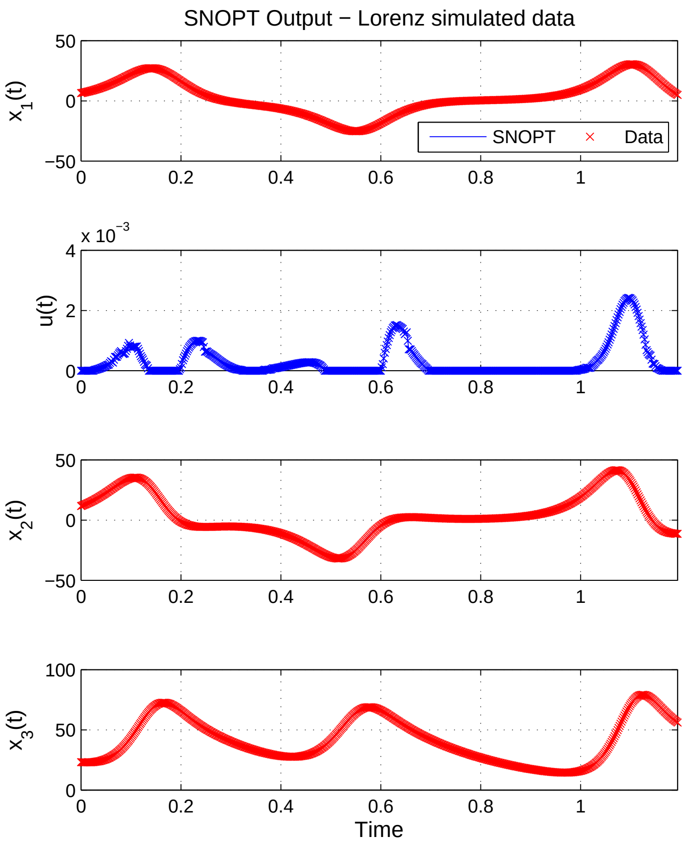


Figure 5.8: SNOPT solution : Lorenz system simulation

Table 5.2: Parameters for Hodgkin–Huxley simulation : constant $I_{\text{inj}}$

| | Data | SNOPT | | Data | SNOPT |
|---|---|---|---|---|---|
| $g_K$ | 6.0 | 6.8782 | $V_K$ | -100 | -101.3056 |
| $g_{Na}$ | 20 | 20.3455 | $V_{Na}$ | 50 | 49.9943 |
| $g_l$ | 0.03 | 0.03381 | $V_l$ | -50 | -52.7179 |
| $a_1$ | 0.01 | 0.009406 | $a_2$ | 50 | 51.1449 |
| $a_5$ | 35 | 34.8983 | $a_4$ | 0.1 | 0.09886 |
| $a_7$ | 0.07 | 0.05003 | $a_8$ | 0.05 | 0.0485 |
| $b_1$ | 0.125 | 0.1456 | $b_2$ | 0.0125 | 0.009871 |
| $b_4$ | 0.0555 | 0.0553 | $b_6$ | 0.1 | 0.09895 |

ically. Repeated data actually adds no new information. It would be better to get the system under study to fire in a non–periodic fashion in order to explore a larger region of the system's phase space. This is exactly what is shown in Figure 5.10. Here, the injected current was taken as a known portion of the Lorenz $x_1(t)$ time–series. This results in non–periodic neuron activity and a dataset that contains more information about the dynamics of the system under study. As can be seen in Table 5.3, the parameter error has been greatly reduced.

Table 5.3: Parameters for Hodgkin–Huxley simulation : Lorenz $I_{\text{inj}}$

| | Data | SNOPT | | Data | SNOPT |
|---|---|---|---|---|---|
| $C_m$ | 1.0 | 0.99975 | $g_K$ | 36 | 35.64223 |
| $V_K$ | -12 | -12.00226 | $g_{Na}$ | 120 | 120.914 |
| $V_{Na}$ | 115 | 115.011 | $g_l$ | 0.3 | 0.33066 |
| $V_l$ | -10.316 | -10.62079 | $a_1$ | 0.01 | 0.01006 |
| $a_2$ | 10 | 9.94615 | $a_3$ | 0.1 | 0.10048 |
| $a_4$ | 0.1 | 0.09952 | $a_5$ | 25 | 24.87718 |
| $a_6$ | 0.1 | 0.10015 | $a_7$ | 0.07 | 0.06974 |
| $a_8$ | 0.05 | 0.04964 | $b_1$ | 0.125 | 0.12488 |
| $b_2$ | 0.0125 | 0.01244 | $b_3$ | 4.0 | 4.00130 |
| $b_4$ | 0.0555 | 0.05542 | $b_5$ | 1 | 1.00298 |
| $b_6$ | 0.1 | 0.09933 | $b_7$ | 30 | 30.14122 |

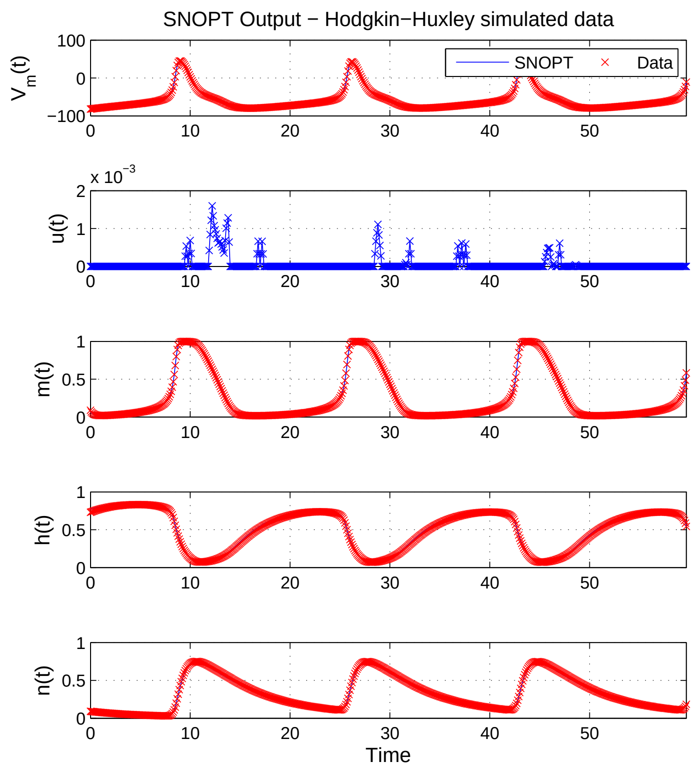


Figure 5.9: SNOPT solution : Hodgkin–Huxley system simulation

## 5.5 Electrical Circuit Experiments

### 5.5.1 Lorenz circuit

Here a model Lorenz system was controlled to track data from a Lorenz electrical circuit with presumed unknown parameters through coupling with the $\mathbf{x}_1(t)$ state. In Figure 5.11 the plots of the data system and the SNOPT output are identical for the observed state $\mathbf{x}_1(t)$ but the unobserved states are not synchronized. This is typical of model error and is most likely due errors in the electrical circuit model itself. The circuit parameters are truly unknown in this case, depending on the exact value of many individual circuit components. Still, the SNOPT output gave parameters within a reasonable range of the designed (guess) values

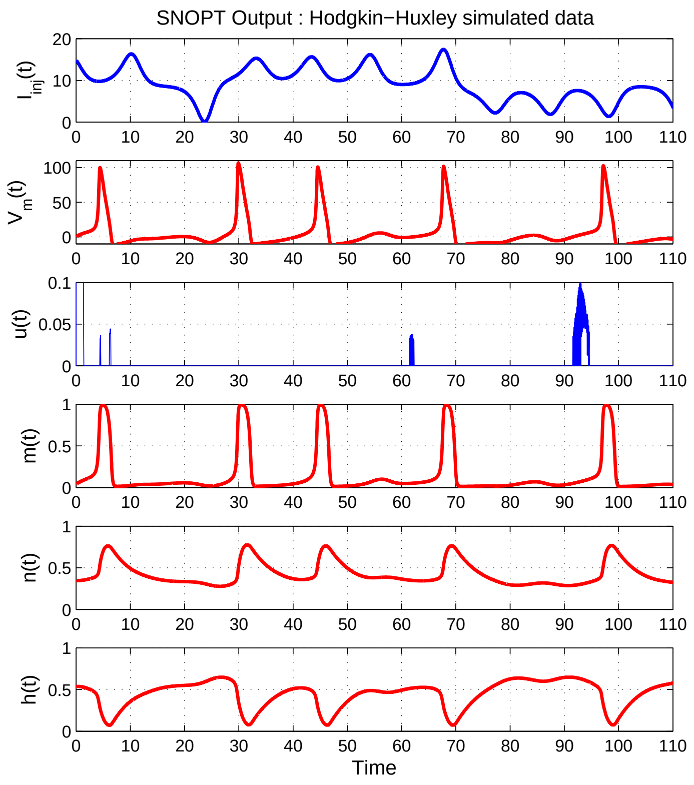


Figure 5.10: SNOPT solution : Hodgkin–Huxley system simulation with Lorenz injection current

(Table 5.4). In this case, due to the speed at which the circuit is running, the control $u(t)$ should be divided by a factor of 1000 for comparison with the Lorenz simulation above. This would give a control magnitude on the order $10^{-1}$ compared to $10^{-3}$ from simulated data. The larger control requirement may be used as an indicator of possible errors in the guessed form of the model. This is an advantageous result since in a real experiment the quantities $x_2(t)$ and $x_3(t)$ will not be measured and therefore it will not be known if those states are actually synchronized with the model.

Table 5.4: Parameters for Lorenz circuit

| | Guess | SNOPT |
|---|---|---|
| $\sigma$ | 16 | 19.67 |
| R | 50 | 56.46 |
| b | 4 | 3.04 |

### 5.5.2 Hodgkin–Huxley circuit

An electrical circuit shown in Figure 5.12 developed at the Institute for Non–Linear Science was designed to emulate a Hodgkin–Huxley neuron. This electrical system was used as the data source and the node corresponding to the cell membrane potential $\mathrm{V}_m(t)$ was coupled into a model Hodgkin–Huxley system. This circuit approximates the rate functions $\alpha_{m,n,h}(\mathrm{V}_m)$ and $\beta_{m,n,h}(\mathrm{V}_m)$ as piecewise linear functions over the range of interest. Every potentiometer, blue rectangular circuit elements in Figure 5.12, needed to be calibrated in order to have known expected values for the model parameters. This calibration was done by using electrical circuit simulation software (SPICE) to determine the potentiometer settings required for specific parameter values (code in Appendix C). These values were then fine tuned in the lab to get the best possible fit for the desired functions $\alpha_{m,n,h}(\mathrm{V}_m)$ and $\beta_{m,n,h}(\mathrm{V}_m)$ as shown in Figure 5.13.

Figure 5.14 shows the SNOPT output from tracking a Hodgkin–Huxley model neuron to the electrical circuit data. Table 5.5 is a listing of the parameters as calibrated in the circuit and as obtained from SNOPT. Parameters are referenced to equations (5.7). Due to the piecewise linear approximations mentioned above, the circuit is known to not exactly reproduce a Hodgkin–Huxley type signal. This is a very encouraging result producing parameters of reasonable value considering the periodic nature of the spiking activity as discussed above.

$$\begin{aligned}
\alpha_n(V_m) &= \frac{n_1(n_2 - V_m)}{e^{n_6(n_2-V_m)} - 1} & \beta_n(V_m) &= n_3 e^{-(n_4+V_m)/n_5} \\
\alpha_m(V_m) &= \frac{m_2(m_1 - V_m)}{e^{m_2(m_1-V_m)} - 1} & \beta_m(V_m) &= m_5 e^{-(m_3+V_m)/m_4} \\
\alpha_h(V_m) &= h_1 e^{-(h_2+V_m)/h_3} & \beta_h(V_m) &= \frac{1}{e^{h_5(h_4-V_m)} + 1}
\end{aligned} \tag{5.7}$$

Table 5.5: Parameters for Hodgkin–Huxley circuit

| | Circuit | SNOPT | | Circuit | SNOPT |
|---|---|---|---|---|---|
| $\mathrm{m}_1$ | 25 | 23.0 | $\mathrm{m}_2$ | 0.1 | 0.0785 |
| $\mathrm{m}_3$ | 0 | -0.300 | $\mathrm{m}_4$ | 18 | 16.2 |
| $\mathrm{m}_5$ | 4 | 6.54 | $\mathrm{h}_l$ | 0.07 | 0.120 |
| $\mathrm{h}_2$ | 0 | 0.00 | $\mathrm{h}_3$ | 20 | 20.8 |
| $\mathrm{h}_4$ | 30 | 23.1 | $\mathrm{h}_5$ | 0.1 | 0.0355 |
| $\mathrm{n}_1$ | 0.01 | 0.00492 | $\mathrm{n}_2$ | 10 | 8.30 |
| $\mathrm{n}_3$ | 0.125 | 0.0528 | $\mathrm{n}_4$ | 0 | 0.00173 |
| $\mathrm{n}_5$ | 80 | 74.6 | $\mathrm{n}_6$ | 0.1 | 0.115 |

## 5.6 Model Verification

The optimal tracking formulation has a very desirable benefit over the other techniques described above. In addition to the parameters and states being estimated, the optimal control required to track the model system to the data is available. Knowledge of this control allows for the comparison of candidate models to one another, ranking which are more or less likely to correctly describe the dynamics of a physical system. One method of using this information is to simply compare the magnitude of control $u(t)$ required for the model to synchronize with data. Associating a need for large coupling strength to model inadequacy.

One interesting example is from the Hodgkin–Huxley electrical circuit experiment discussed above. During one particular exercise there was an error in the Python code that describes the neuron model. In particular, $\beta_n(\mathrm{V}_m)$ was given the incorrect form. As can be seen in Figure 5.15, observing the model synchronization to the dataset, $\mathrm{V}_m$, alone may lead to the conclusion that the parameter and state estimation result was successful. By looking at the value of the control required for the model to track the data we are able to immediately notice that there may be an error in the model. The optimal control is two orders of magnitude greater than the example above with the correct model form (Figure 5.14).

Another method of using the information contained in the optimal control function $u(t)$ is to require that the magnitude of the feedback term be small compared to the dynamics of the model. Consider equation (5.2) above. The dynamics of the measured state is $\frac{dy_1(t)}{dt} = F_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q}) + u(t)(x_1(t) - y_1(t))$. We require that the feedback term, $u(t)(x_1(t) - y_1(t))$, be much smaller than the governing model dynamics, $F_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q})$. This would ensure that the model is evolving primarily based on its own dynamics with only weak influences from the control to overcome tracking problems associated with noise or complex dynamics. Consider the ratio of the square of the two quantities, model and model plus feedback.

$$R(t) = \frac{[\mathrm{F}_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q})]^2}{[\mathrm{F}_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q})]^2 + [u(t)(x_1(t) - y_1(t))]^2}$$

This ratio is equal to one when the dynamics is governed by the model term, $F_1(y_1(t), \mathbf{y}_\perp(t), \mathbf{q})$, and tends to zero as the coupling term, $u(t)(x_1(t) - y_1(t))$ begins to dominate the time evolution of the system. We may now compute $R(t)$ of several candidate models driven to track the same data set and look at this ratio as an indicator of how well each model is able to track the data on its own, considering the 'best' model to be that with an $R(t)$ staying nearest to one. An example is shown in Figure 5.16 and Figure 5.17. Here the model was assumed to be the Lorenz system (Eq. 2.2) and the data sets were from a Lorenz system and the modified Lorenz system shown below with an additional term, 5X, in the $\dot{\mathrm{Z}}$ equation.

$$\begin{aligned}
\dot{\mathrm{X}} &= \sigma(\mathrm{Y} - \mathrm{X}) \\
\dot{\mathrm{Y}} &= \mathrm{RX} - \mathrm{Y} - \mathrm{XZ} \\
\dot{\mathrm{Z}} &= \mathrm{XY} - \mathrm{bZ} + 5\mathrm{X}
\end{aligned}$$

Figure 5.16 shows the magnitude of the control and the ratio $R(t)$ defined above for the case of matched model and data systems. The model synchronizes with the data, the control is on the order $10^{-1}$ and the ratio $R(t)$ remains near one. Figure 5.17 shows results when a Lorenz model is introduced to data from the modified Lorenz system above. The Lorenz model is still tracking the data, although not as well as in Figure 5.16, but the control is order one and the ratio $R(t)$ takes several diversions from near one.

The optimal control function $u(t)$ allows a model to track data sets in the presence of complex dynamics, measurement noise, and incorrect terms in the model dynamics. Provided that candidate models are introduced to identical data sets, so that system and measurement noise are identical, any additional inability of the model to track the data is captured in variations among the optimal control functions $u(t)$ for each model. This ability to develop metrics to compare candidate models to one another based on this control function $u(t)$ is a very powerful and important aspect of the optimal control method of parameter and state estimation.

## 5.7 Time-Dependent Parameters

There are times when one has parameters in a system whose time dependence is dictated by external forces and is not governed by a differential equation in the model. If there were a differential equation for the parameters, such parameters would become equivalent to and included in the state variables $\mathbf{y}(t)$. The ability of the optimal control approach to handle such a situations in the Colpitts oscillator system is briefly explored. We will allow for the parameter $\alpha$ to be a function of time, $\alpha(t)$, while keeping the remaining parameters as the known fixed values $\gamma = 0.08$, q $= 0.7$ and $\eta = 6.3$.

It is a simple task to allow a parameter to vary at each co–location point. We may allow the parameter $\alpha$ in the Colpitts system to take on any positive value at each point, effectively trading a single parameter $\alpha$ for M parameters $\alpha(m)$ where M is the number of co–location points that define the optimization problem to be solved. First, the model is asked to track data from a system with constant parameter $\alpha = 5.0$. The resulting values for $\alpha(m)$ are shown in Figure 5.18. In general the SNOPT solution is tending to keep the parameter near its correct value, but the addition of M$-1$ unknown variables without any additional constraints has compromised the ability of this method to find the particular optimized solution that we are looking for. Figure 5.19 is a repeat of the same calculation above but allowing the parameter in the data system to be a more complicated function, $\alpha(t) = 5 + 4 \cdot \tanh((t - 50)/25)$. In this case the SNOPT estimate of the parameter values $\alpha(m)$ seem to track the actual value better than in the case of $\alpha(t) = 5.0$ above. This is likely due to the data system never being allowed to rest on an attractor, it is instead constantly in a transient state with a data time–series more rich in information than the case of constant $\alpha$. This is similar to the above case of exciting the Hodgkin–Huxley model with a Lorenz signal to force a larger exploration of phase space. This, too, resulted in improved performance of the optimal tracking method.

Instead of allowing the parameter to vary at each co–location point without constraint, we could introduce the constraint of requiring the parameter to fit a specific function of time. Since we have no knowledge of the actual time variation from the observations of $x_1(t)$ we may impose a smoothness criterion on $\alpha_D(t)$ in the form of a polynomial representation. In particular we asked that a cubic polynomial describe the time variation of $\alpha_D(t)$ over an interval $K\tau$ where $K$ is selected by us. In the case presented here, the result did not depend on $K$ over a range $25 \leq K \leq 200$. Figure 5.20 shows results for $K = 25$. The time dependence selected was chosen to have $\alpha_M(t)$ vary from near one to as large as 8 or 9 thus crossing the boundaries of the bifurcation sequence where the Colpitts oscillator goes from a fixed point attractor through limit cycles to chaotic oscillations. In particular, the following function was selected.

$$\alpha_D(t) = 5 + 4e^{(t-100)^2/2000} \sin(2\pi t/50)$$

The initial conditions were determined by the optimization, and all state variables tracked the data. The ability of the optimal control method combined with the SNOPT optimization package to allow for time dependent parameters is very powerful and encouraging, especially if one is interested in modeling a network that is known to change its connectivity strengths over time.

## 5.8 Adaptive Grid

This discussion has concentrated on discretizing a system using co–location points equally spaced in time. This evenly spaced grid of points allows for simplified constraint equations and therefore simplified computer code. One of the advantages of the Python front–end is that it allows a user to easily generate an optimization problem with arbitrarily spaced co–location points. In some cases, the physical system under study may exhibit dynamics on widely separated time scales. The slow dynamics require a long sample time and the fast dynamics require a fast sample rate. The combination of these two requirements makes the total number of co–location points necessary to capture all of the dynamics extremely large. In these cases, the ability to adaptively determine the co–location points is desirable.

Here we will take the approach of starting with a sparsely populated grid and add points when too large of an error is detected in the integration rule from one co–location point to the next. A Python application was written with the ability to define the optimization problem using an arbitrary number of data points. A data file is read in along with a corresponding boolean array of the same size. If the boolean associated with a data point is TRUE, that data point will be used in the optimization problem, otherwise it will not be used. A sample SNOPT run using the Hodgkin–Huxley neuron model is shown in Figure 5.21. In this case, points were predetermined based on the derivative of the measured data. Only 143 of the total 800 points were needed to reproduce similar results as shown above. This drastically reduces the problem size and increases the speed of computation.

$$\begin{array}{rl} \text{Data file} \rightarrow & \left[\begin{array}{cccccc} 2.34 & 1.09 & -0.34 & -2.54 & \cdots & 7.24 \end{array}\right] \\ \text{Boolean array} \rightarrow & \left[\begin{array}{cccccc} \text{True} & \text{False} & \text{False} & \text{True} & \cdots & \text{True} \end{array}\right] \end{array}$$

Since we will have control over accuracy by the ability to place co–location points, the trapezoid integration method $\mathrm{y_{n+1} = y_n + \frac{h}{2}\left[F(y_n) + F(y_{n+1})\right]}$ will be used to generate constraints for the optimization problem. A smoothness requirement still exists on the control in the form of satisfying a cubic polynomial. In other words, control $U(n+1)$ is constrained to fit a cubic polynomial determined from $U(n)$, $dU(n)$,$U(n+2)$ and $dU(n+2)$. Since the interior point on this cubic will not in general be the center point, the general form of the cubic constraint (5.6) was used with $s = h(n)/(h(n) + h(n+1))$ where $h(n) = t(n+1) - t(n)$ is the amount of time between points $n$ and $n+1$.

A rule is needed to determine if a new co–location point must be added between $n$ and $n+1$. Here we will estimate the integration error of the trapezoid rule (a $2^{nd}$ order method) using a higher order method, in this case the $4^{th}$ order Runge–Kutta method. Figure 5.22 depicts the strategy of adding co–location points.

We begin by starting with an evenly spaced, sparse grid of co–location points. The SNOPT solver is then run for several tens of major iterations. Each state from the output $\mathbf{Y}(n)$ is then integrated forward in time using Runge–Kuta and the resulting state $\mathbf{Y}_{rk4}(n+1)$ is compared to the SNOPT value $\mathbf{Y}_{trap}(n+1)$. If the resulting error magnitude $|y_{rk4}(n+1) - y_{trap}(n+1)|$ is larger than some predetermined tolerance then a co–location point is added near the midpoint of the interval. The Python code then generates a new optimization problem based on the new grid points and initial conditions from the previous solution. This process is repeated until no new points are required to be added. The Figures 5.23 and 5.24 demonstrate this process applied to the Hodgkin–Huxley neuron model. The process begins with the upper left plot of Figure 5.23 with a grid of 42 evenly spaced grid points. Red coloring indicates that the next iteration will divide the interval to the right in half. Blue coloring indicates that the error is within bounds and the interval to the right does not need to be shortened. The solution ends at the bottom left plot of Figure 5.24 with a total of 203 co–location points out of a dataset of 800 points. The bottom right plot of Figure 5.24 includes the final estimate of the integration error for each of the state variables.

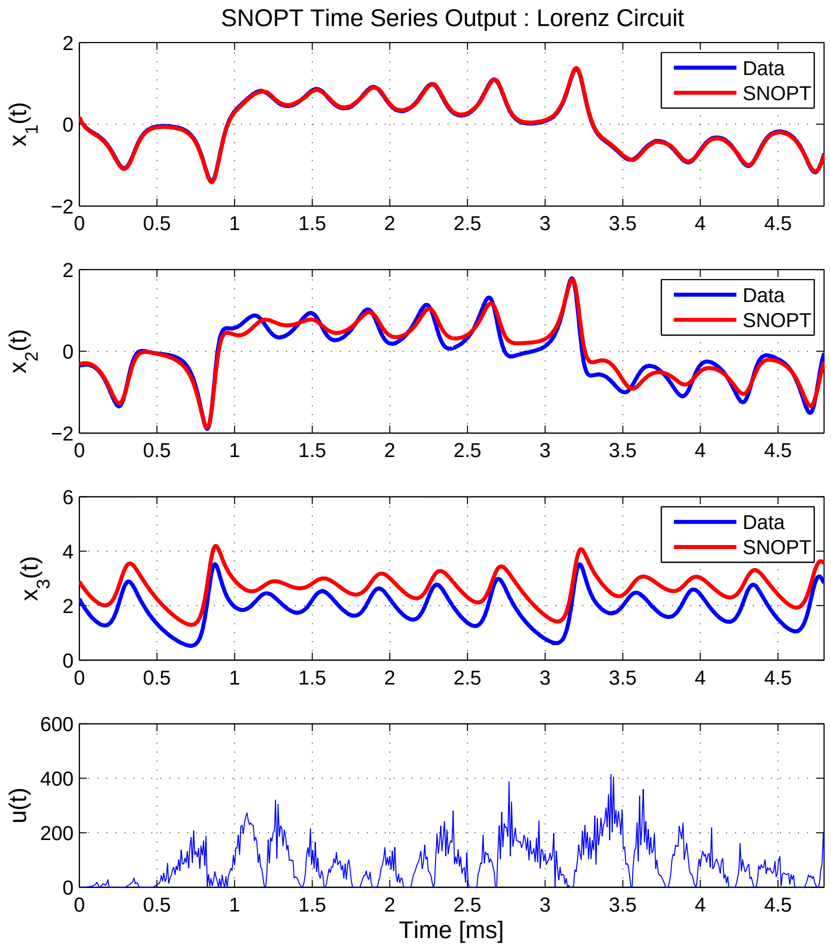


Figure 5.11: SNOPT solution : Lorenz system circuit

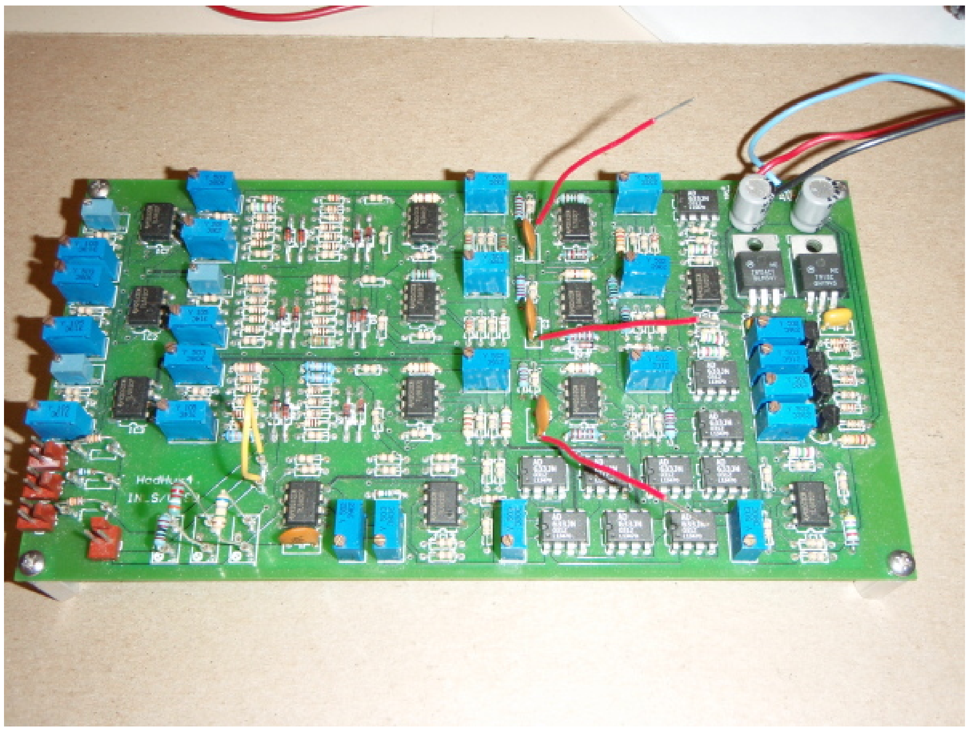


Figure 5.12: Hodgkin–Huxley circuit board

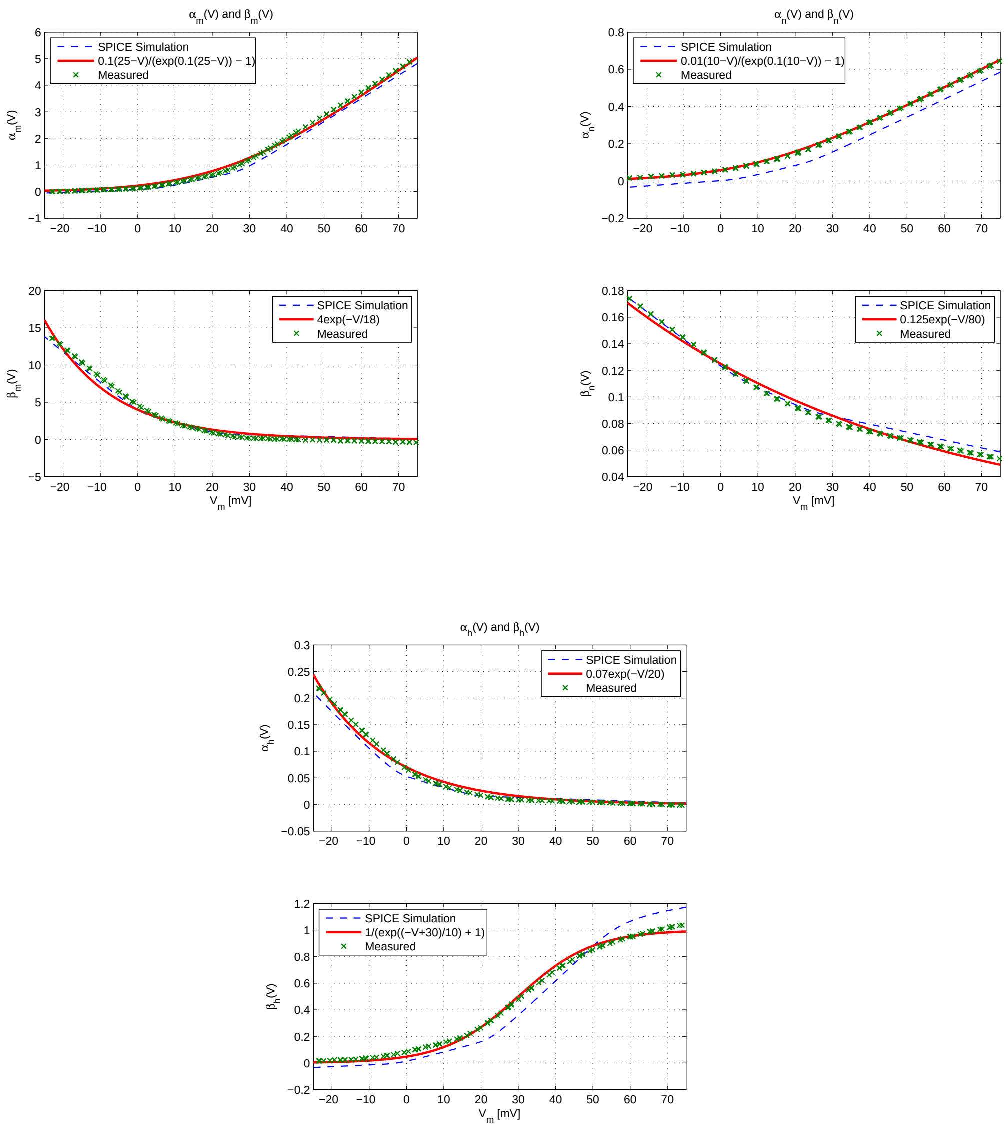


Figure 5.13: Circuit $\alpha_{m,n,h}(\mathrm{V}_m)$ and $\beta_{n,m,h}(\mathrm{V}_m)$

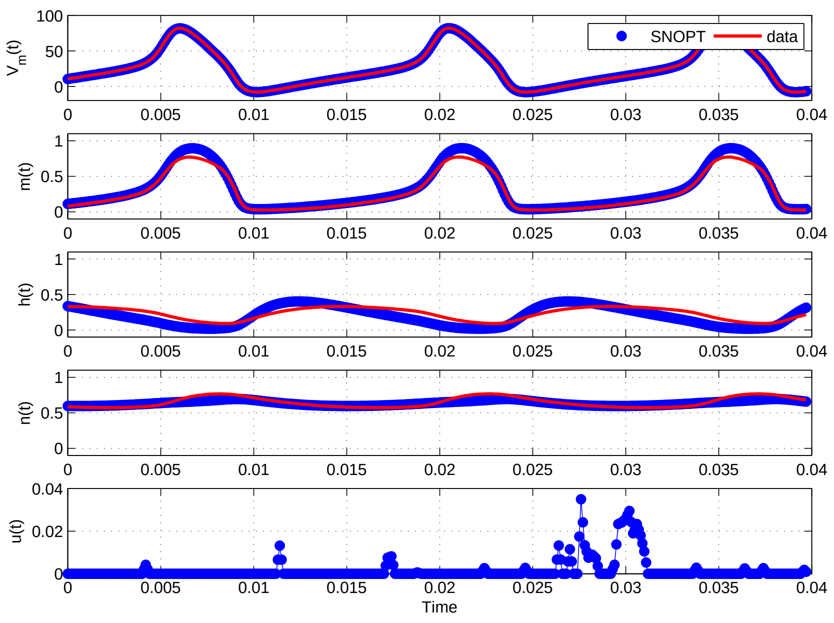


Figure 5.14: Hodgkin–Huxley circuit SNOPT output #1

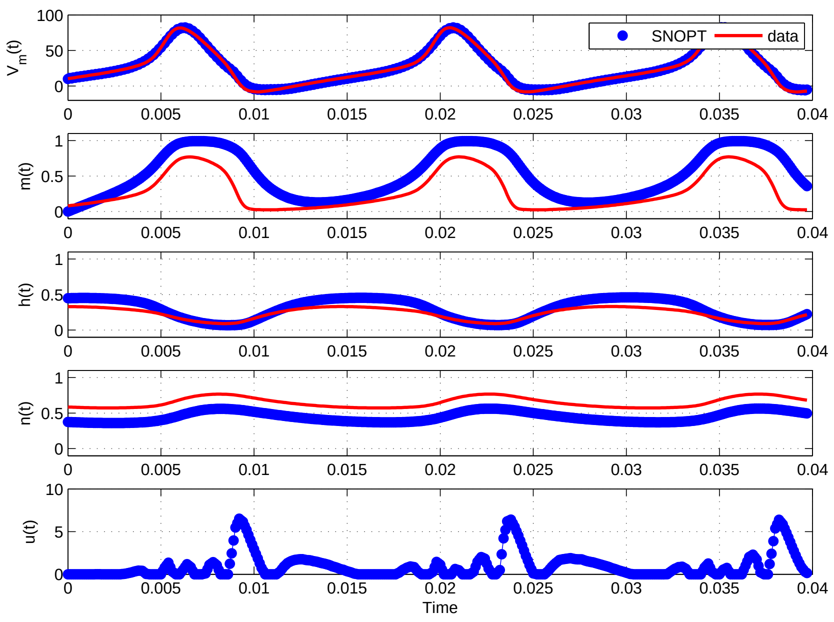


Figure 5.15: Hodgkin–Huxley circuit SNOPT output #2

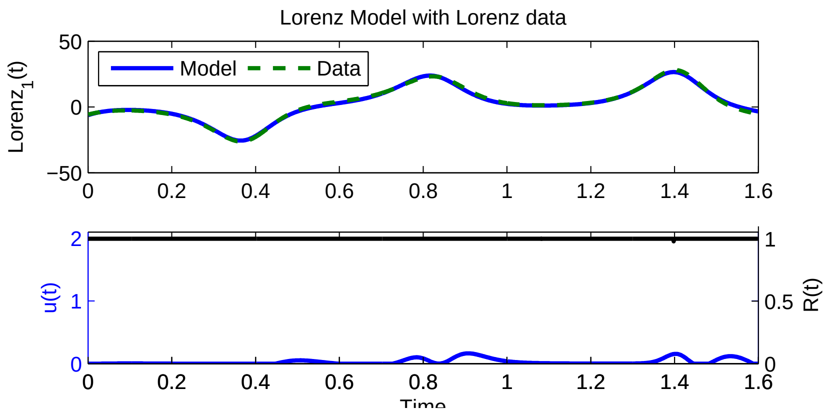


Figure 5.16: Lorenz : Correct model

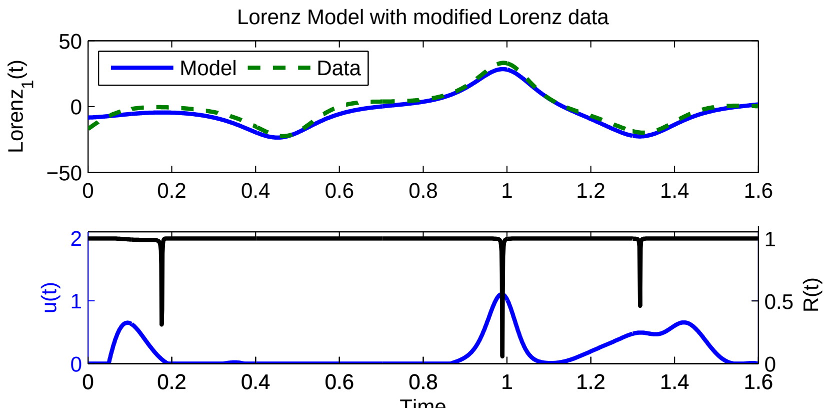


Figure 5.17: Lorenz : Incorrect model

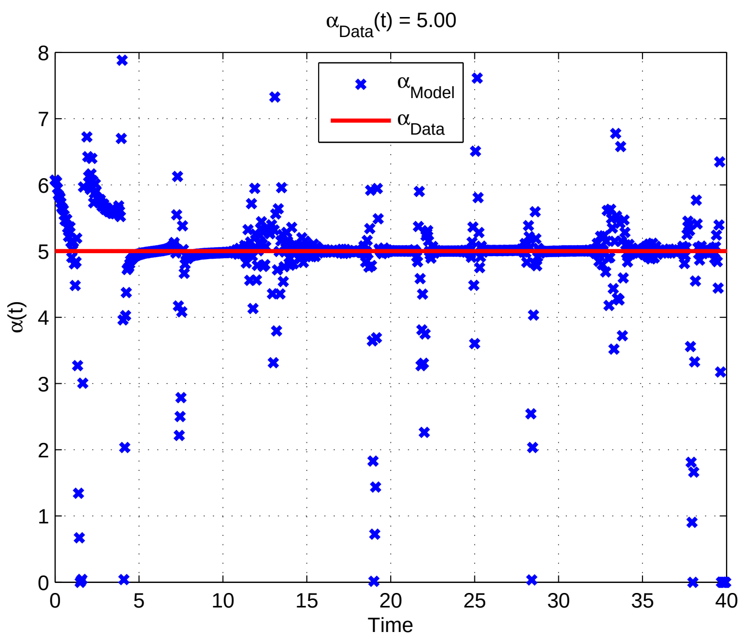


Figure 5.18: Colpitts : $\alpha(t)$ solution #1

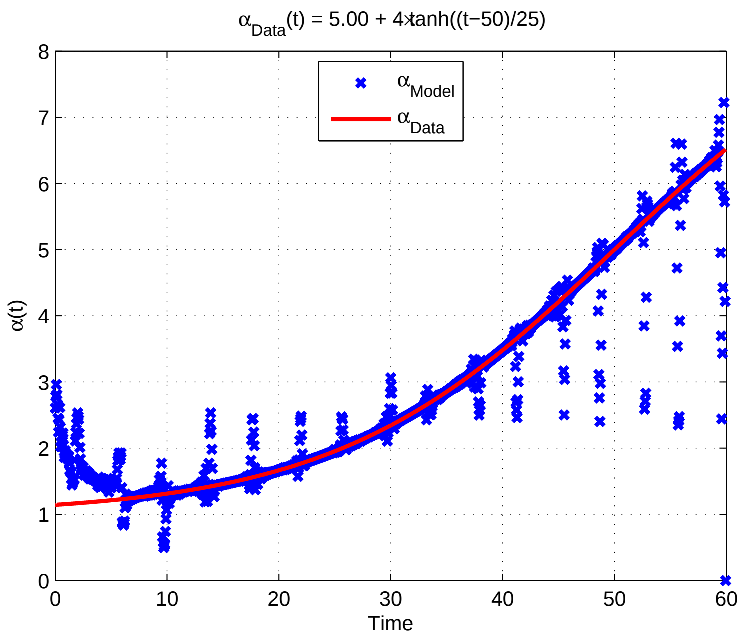


Figure 5.19: Colpitts : $\alpha(t)$ solution #2

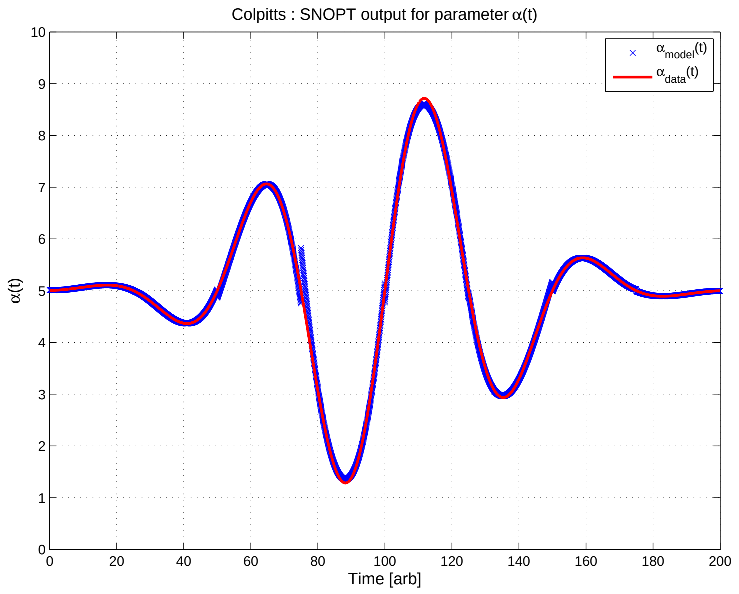


Figure 5.20: Colpitts : $\alpha(t)$ solution #3

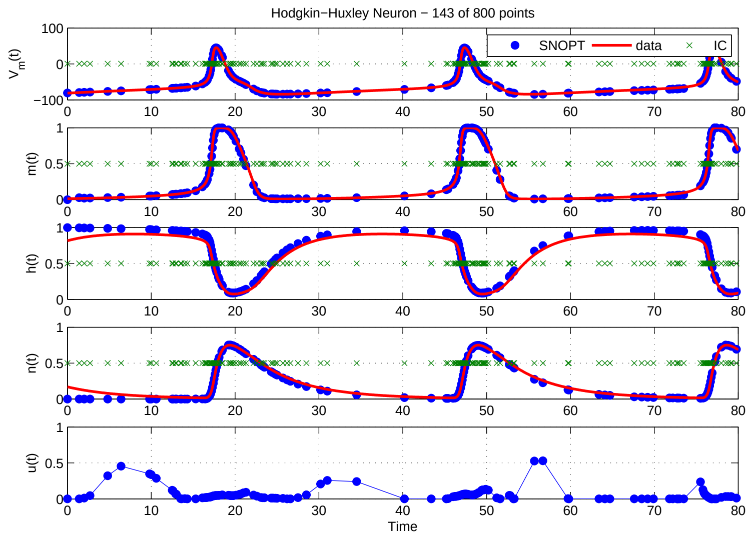


Figure 5.21: Test of arbitrary co–location assignment

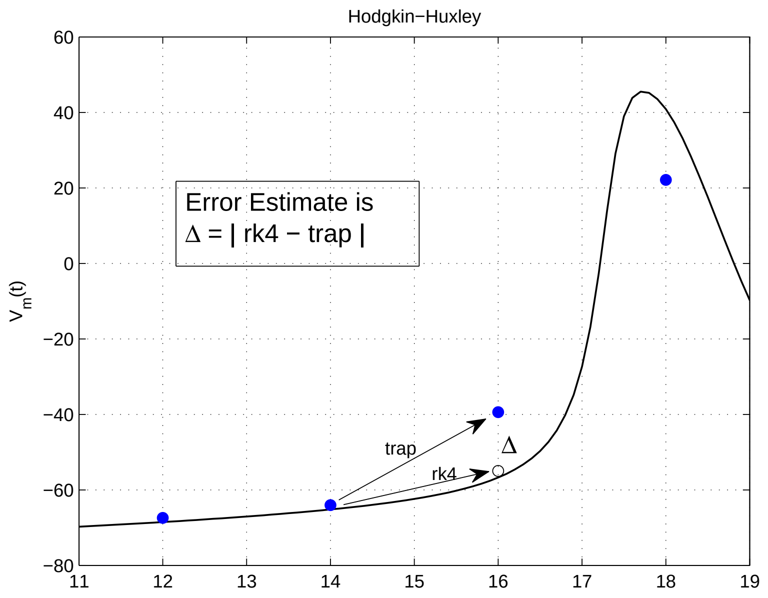


Figure 5.22: Adaptive co–location rule

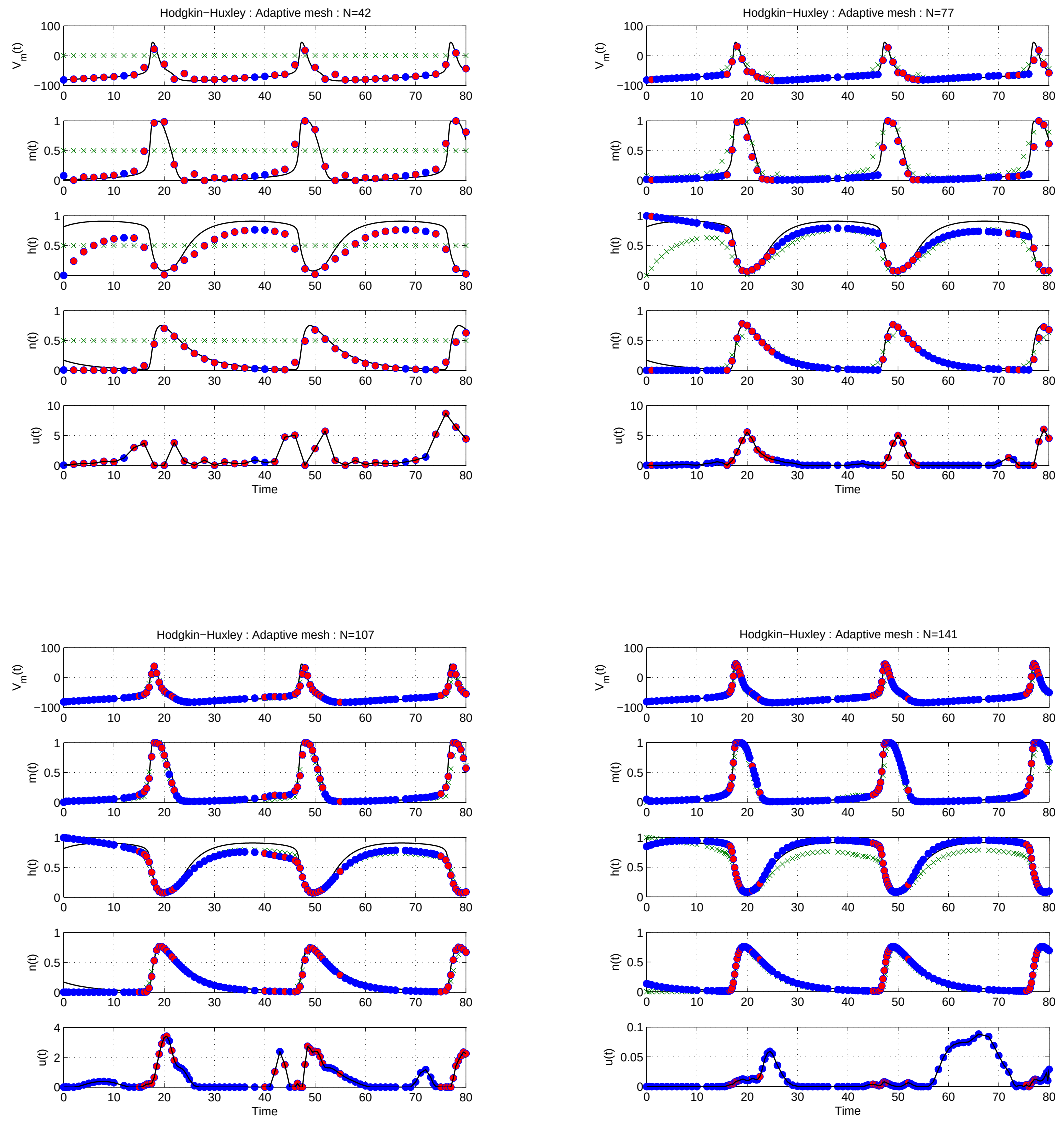
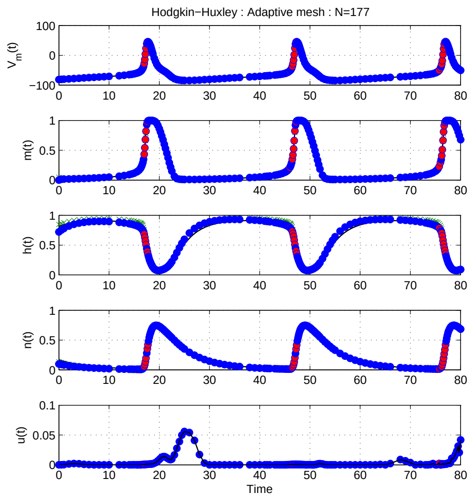
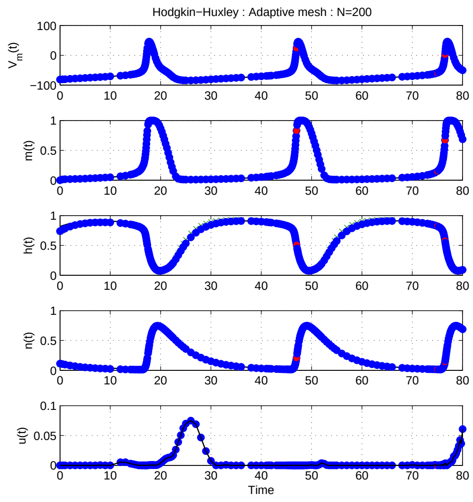
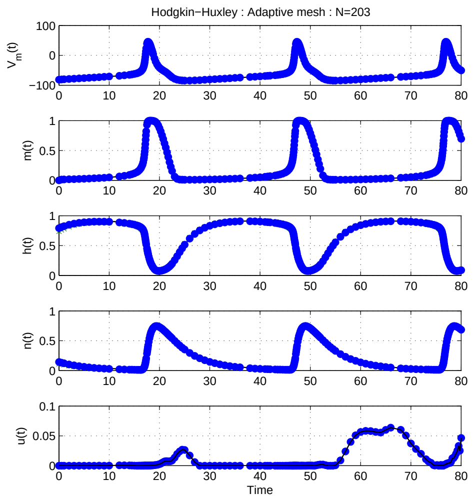


Figure 5.23: Hodgkin–Huxley adaptive mesh solution (Part I)







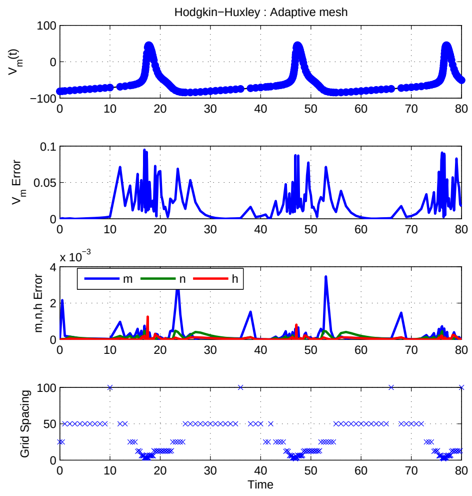


Figure 5.24: Hodgkin–Huxley adaptive mesh solution (Part II)

# Chapter 6

# Conclusion

Determining unknown model parameters from observed data is one of the critical and traditional steps in developing predictive models [39]. The idea of using synchronization of the data source and the model to establish unknown parameters is not at all a new idea, and in one manner or another has always been used in this context [30]. The 'synchronization' in an automated fashion as explored in this paper has not always been the established procedure where 'fits good to the eye' or other qualitative methods have been used. In trying to implement formal synchronization of the experimental system producing the data and a proposed model for use in predicting the future behavior of the system, a problem is encountered where if the coupling of the data into the dynamical equations of the model is too large, the variation of a traditional, least squares cost function suffers from very weak variations in the desired parameters, thus reducing the value of the method for determining those parameters. Also, if the coupling is too weak and the system is chaotic, the very instabilities that lead to the chaos interfere with parameter determination through a numerically unstable cost function[30]. This dilemma has led to the three forms of 'balanced synchronization' which have been discussed and explored in this thesis.

First, in addition to the least square cost or other comparison of the data input and equivalent model output, an additional cost is added associated with the largest conditional Lyapunov exponent (CLE) of the model, conditioned on being driven by the data. If one requires this CLE to be slightly negative, this bounds the value of the coupling while assuring the data and the model remain synchronized. When this synchronization occurs, information about the parameters contained in the data is efficiently passed to the model. Any less information and the model will no longer synchronize to the data causing the cost function to develop many local minima. Any more information and the cost function becomes too flat causing the location of the minimum to be numerically uncertain. One drawback of this method is that the largest CLE may be very costly to calculate, especially in the presence of noisy measurements.

The second method solves the problem of calculating the largest CLE by augmenting the dynamics of the model system adding a temporal variation of the coupling between the data and the model. The dynamics for the coupling is motivated by the desire for the coupling to be as small as possible but allowing for the coupling to increase when the synchronization error grows. The proposed coupling dynamics balances the driving of the coupling to zero against the mismatch of the data and model signals. This method allows for the augmented model system and the data source to exhibit generalized synchronization. The augmented model system exhibits spiking behavior in the synchronization error and motivates a new form of cost function. The 'cost' becomes the probability of observing the model and system to be **un**synchronized at any particular point in time. This cost function has a well defined minimum associated with the model parameters equivalent to the data system parameters. This cost function, being essentially a low pass filter, also remains smooth in the presence of measurement noise.

The last method expanded on the first and reformulated the problem of parameter and state estimation as an optimal tracking problem. This idea used a direct transcription method for posing the model dynamics as an optimization problem and allowing the data coupling parameter to take on values at each co–location point. Where in the second method above, the coupling was constrained to follow a differential equation in time, here the coupling may be an arbitrary smooth function of time imposed through a Hermite interpolation requirement. We seek the model parameters that allow for the minimal amount of control needed to track the data. This control function, called $u(t)$, may then be used as a measure to compare candidate models against one another.

These three methods developed in this thesis are meant to be new tools for the parameter and state estimation toolbox, augmenting existing techniques. It was also intended to bring some techniques into play that are developed on a foundation of nonlinear system dynamics from the beginning instead of patches applied to ideas originating from linear system theory. It is my hope that the methods developed here will provide a useful tool for the study and validation of complex mathematical models of physical systems. Of course, the method does not replace the need to develop these models based on keen insights into the physics of the processes involved.

# Appendix A

# Further Work

The more one studies the problem of parameter and state estimation the more options for further extensions come to mind. This appendix briefly introduces two topics aimed at extending the capabilities of the methods introduced in this thesis. They are, in my opinion, deserving of further examination.

## A.1 Space-Time Adaptive Grid

Section 5.7 introduced a method of adapting the position of co–location points in order to couple more information to the model when required, while allowing for more sparse co–location points in regions where the dynamics is less demanding. For the same reasons, in cases where the phenomenon is described by a partial differential equation it would be desirable to have spatial grid control as well. An example of current interest is the model my Ananthakrishna [5, 4] describing spatio–temporal behavior of dislocations in dilute metallic alloys under a range of strain rates and temperatures know as the Portevin–Le Chatelier (PLC) effect [9]. The model equations are shown below with $\phi_{eff} = (\phi - h\rho_{im}^{1/2})$.

$$\begin{aligned}
\frac{\partial \rho_m(x,t)}{\partial t} &= -b_o\rho_m^2 - \rho_m\rho_{im} - a\rho_m + \phi_{eff}^m\rho_m + \frac{D}{\rho_{im}}\frac{\partial^2(\phi_{eff}^m\rho_m)}{\partial x^2} \\
\frac{\partial \rho_{im}(x,t)}{\partial t} &= bo(b_o\rho_m^2 - \rho_m\rho_{im} - \rho_{im} + a\rho_c)
\end{aligned}$$

$$\begin{aligned}\frac{\partial \rho_c(x,t)}{\partial t} &= c(\rho_m - \rho_c) \\ \frac{\partial \phi(t)}{\partial t} &= d\left[\dot{\epsilon} - \frac{1}{l}\int_0^l \rho_m(x,t)\phi_{eff}^m(x,t)\,dx\right]\end{aligned}$$

$\rho_m, \rho_{im}, \rho_c$ are the densities of mobile, immobile, and Cottrell's type dislocations respectively. Numerical integration of this system with typical parameter values of $a = 0.8$, $b_o = 0.0005$, $c = 0.08$, $d = 0.00006$, $m = 3.0$, $h = 0.0$ and $D = 0.5$ results in the complex time–series for the measurable stress $\phi(t)$ shown in Figure A.1. This time–series is comparable, both in magnitude and information content, to stress measurements in laboratory experiments. The resulting mobile dislocation density is shown in Figure A.2. The plot of $\rho_m(x,t)$ shows regions of activity surrounded

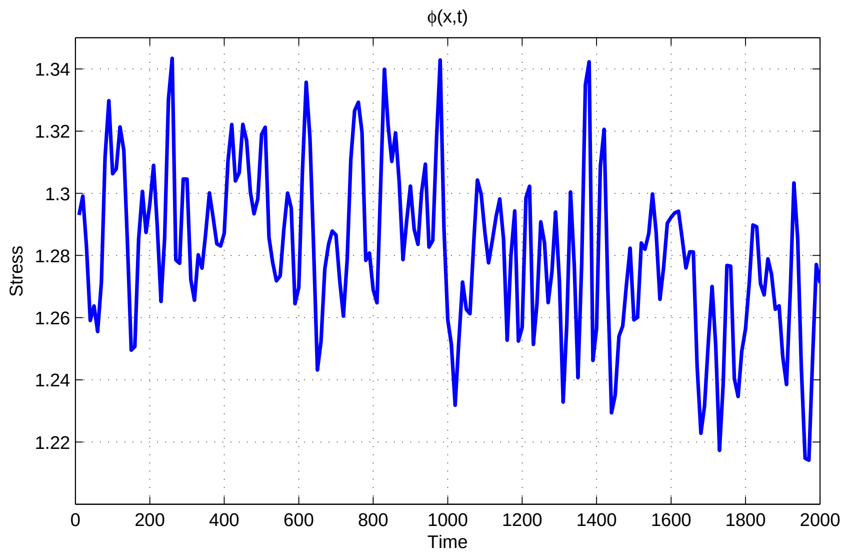


Figure A.1: PLC : Stress time–series

by large regions of inactivity in space–time. It is desirable to modify the methods presented in this thesis to allow for both spatial and temporal control of grid points in order to keep the optimization problem size as small as possible. This is a problem with complex spatio–temporal dynamics with only a single measurable, $\phi(t)$, dependent on the other states $\rho_m(x,t)$, $\rho_{im}(x,t)$ and $\rho_c(x,t)$ in a very complex fashion. The ability of these methods to deduce the parameters of this model from experimental data is important to address. It would test the limits of the

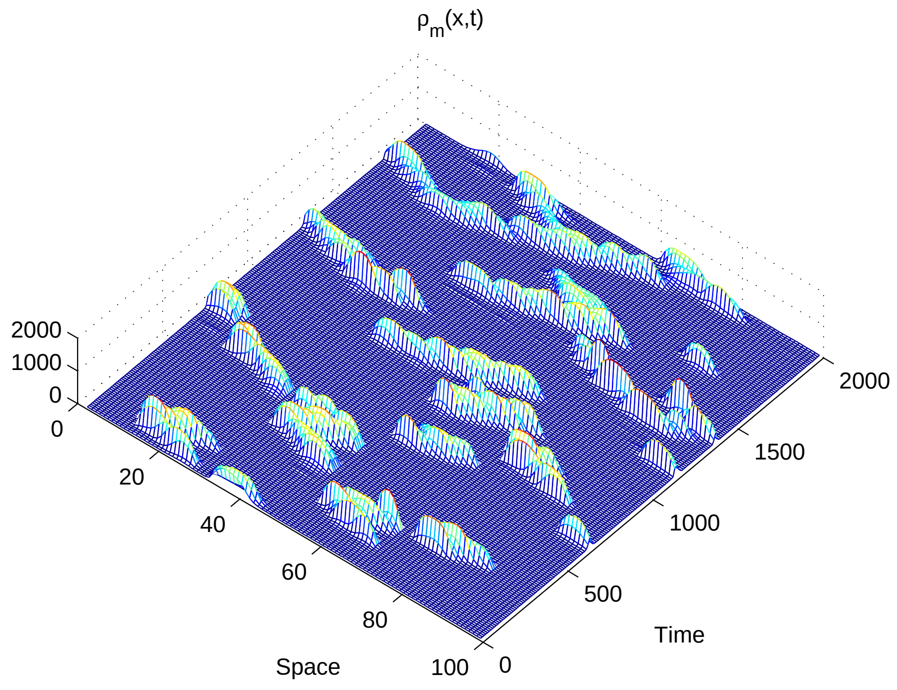


Figure A.2: PLC : Mobile dislocation density

synchronization method described in this thesis while possibly providing some important insight into the PLC phenomenon itself.

## A.2 Time–delay Space Control

This text dealt with example nonlinear systems with three or four state variables. When the problem increases in size (a network of neurons for example) there may be more than one positive Lyapunov exponent requiring control. This requires either coupling more measured states (which may or may not be physically available) to the model or using the available measurements more effectively. The control scheme presented in this thesis addresses coupling data to a model using feedback dependent on an instantaneous measurement. A more effective method of coupling would be to use some history of the measurements. The idea of time–delayed coordinates presented in Chapter 2 provides some guidance on how to proceed.

Given a discretized D–dimensional system of the form $\mathbf{X}(n+1) = \mathbf{F}(\mathbf{X}(n))$ we know that there exists some nonlinear transformation [32] from the phase–space coordinates $\mathbf{X}(n) = [x_1(n), x_2(n), x_3(n), \ldots, x_D(n)]$ to a time–delayed set of coordinates of one of the state variables, say $x_1(n)$. The new time–delayed vector is denoted by $\mathbf{S}(n) = [x_1(n), x_1(n-\tau), x_1(n-2\tau), \ldots, x_1(n-r\tau)]$, where $r$ is at most $2D+1$ and $\tau$ is the unit of time delay. The associated coordinate transformation is denoted by $\mathbf{S}(n) = \Phi(\mathbf{X}(n))$. The states $\mathbf{S}(n)$ evolve in time according to $\mathbf{S}(n+1) = \mathbf{G}(\mathbf{S}(n))$ where the dynamics in the time–delayed coordinate space, $\mathbf{G}$, is related to the dynamics in phase–space, $\mathbf{F}$, by $\mathbf{G} = \Phi \circ \mathbf{F} \circ \Phi^{-1}$. The idea is that all of the states in $\mathbf{S}(n)$ are measurable as opposed to only a limited number of the states in $\mathbf{X}(n)$, allowing for full proportional feedback control of the model. It is instructive to see an example worked out with a linear system; take the damped harmonic oscillator.

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -bx_2 - \omega^2 x_1
\end{aligned}$$

We wish to discretize this system of equations into the form $\mathbf{X}(n+1) = \hat{\mathbf{F}}\mathbf{X}(n)$ and find the transformation to the delayed coordinates $[x_1(n), x_1(n-1)]$. Choosing Euler's method integration for this example, the discretized system becomes

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \begin{bmatrix} 1 & h \\ -h\omega^2 & (1-hb) \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix}$$

Which may be used to solve for the transformation $\hat{\Phi}$ such that $\mathbf{S}(n) = \hat{\Phi}\mathbf{X}(n)$. In this case the transformation is

$$\begin{bmatrix} x_1(n) \\ x_1(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1-hb}{1-hb+(h\omega)^2} & \frac{-h}{1-hb+(h\omega)^2} \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix}$$

Now we may write the dynamics in the time–delayed coordinates by applying this transformation to the equation of motion.

$$\mathbf{X}(n+1) = \hat{\mathbf{F}}\mathbf{X}(n)$$

$$\begin{aligned}\hat{\Phi}\mathbf{X}(n+1) &= \hat{\Phi}\hat{\mathbf{F}}\hat{\Phi}^{-1}\hat{\Phi}\mathbf{X}(n)\\ \mathbf{S}(n+1) &= \hat{\Phi}\hat{\mathbf{F}}\hat{\Phi}^{-1}\mathbf{S}(n)\end{aligned}$$

Observing the fact that all states in this space are observable, we can couple the measured data to the time–delayed system model using a diagonal matrix of coupling gains, $\hat{\mathbf{U}}$, multiplied by the time–delayed error vector $\mathbf{e}(n)$. The final step is a transformation back to the phase–space coordinates $[x_1(n), x_2(n)]$.

$$\begin{aligned}\mathbf{S}(n+1) &= \hat{\Phi}\hat{\mathbf{F}}\hat{\Phi}^{-1}\mathbf{S}(n) + \hat{\mathbf{U}}\mathbf{e}(n)\\ \mathbf{X}(n+1) &= \hat{\mathbf{F}}\mathbf{X}(n) + \hat{\Phi}^{-1}\mathbf{U}\mathbf{e}(n)\end{aligned}$$

In this example, the feedback terms become

$$\begin{aligned}\hat{\Phi}^{-1}\mathbf{U}\mathbf{e}(n) &= \begin{bmatrix} 1 & 0 \\ \frac{1-hb}{h} & \frac{-(1-hb+(h\omega)^2)}{h}\end{bmatrix}\begin{bmatrix} U_{11} & 0 \\ 0 & U_{22}\end{bmatrix}\begin{bmatrix}\mathbf{e}(n) \\ \mathbf{e}(n-1)\end{bmatrix}\\ &= \begin{bmatrix} U_{11} & 0 \\ \frac{1-hb}{h}U_{11} & \frac{-(1-hb+(h\omega)^2)}{h}U_{22}\end{bmatrix}\begin{bmatrix}\mathbf{e}(n) \\ \mathbf{e}(n-1)\end{bmatrix}\\ &= \begin{bmatrix} U_{11}\mathbf{e}(n) \\ U_{11}\frac{1-hb}{h}\mathbf{e}(n) - U_{22}\frac{(1-hb+(h\omega)^2)}{h}\mathbf{e}(n-1))\end{bmatrix}\end{aligned}$$

Where $\mathbf{e}(n) = x_{data}(n) - x_1(n)$ and $\mathbf{e}(n-1) = x_{data}(n-1) - x_1(n-1)$. The coupling to $x_1(n)$ is the usual proportional error coupling $U_{11}(x_{data}(n) - x_1(n))$, while the coupling to $x_2(n)$ includes information from both the current measurement and the previous measurement to estimate the feedback term. In this linear system example the coordinate transformation from phase–space to time–delayed space was know. In the general nonlinear case the transformation $\Phi(X(n))$ will be known to exist but its explicit form will be unknown.

# Appendix B

# Hardware Dynamical Coupling of Two Lorenz Systems

An electrical circuit implementation of the dynamical coupling ($K(t)$) method of parameter estimation was built to explore this method in a more realistic setting. The experimental setup is as shown in Figure(B.1). The data signal $x_1(t)$ is generated from a Lorenz system with fixed parameters ($\sigma_1$,$\mathrm{b}_1$,$\mathrm{R}_1$). The model is a Lorenz system with two parameters ($\sigma_2$ and $\mathrm{b}_2$) identical to the Data system and an adjustable third parameter, $\mathrm{R}_2$. A Data Acquisition (NI-DAQ) card connected to a PC running LabVIEW software acquires $x_1(t)$, $x_2(t)$ and $K(t)$ at a rate of 50kHz. The PC is able to control the value of $\mathrm{R}_2$ by turning a stepper motor that is connected to a 10-turn potentiometer. A typical parameter scan consists setting $\mathrm{R}_2$, taking 2 seconds of data, computing the value of the cost function and repeating over the available range of $\mathrm{R}_2$ (in particular, $23.5 \leq \mathrm{R}_2 \leq 50.6$).

The basic design of the Lorenz portion of the electrical circuit is adapted from Cuomo [7]. In order to keep voltages in the range of the available power supplies, all state variables are scaled by a factor of 20. Time is scaled by a factor of 1000. With these scalings, the Lorenz system that needs to be realized in the electrical circuit is the following:

$$\dot{x} = 1000[\sigma(y-x)]$$

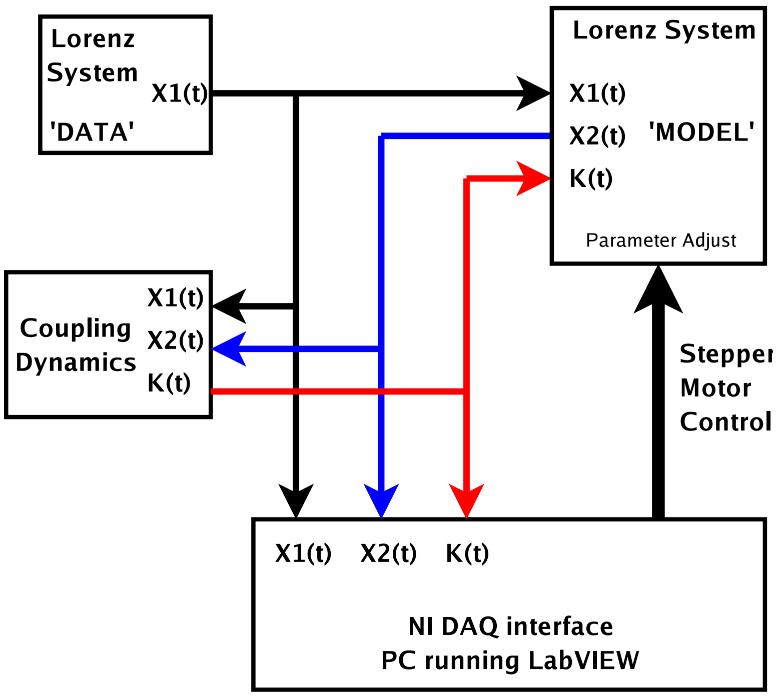


Figure B.1: Diagram of experimental setup.

$$\begin{aligned}\dot{y} &= 1000[\mathrm{R}x - y - 20xz] \\ \dot{z} &= 1000[20xy - \mathrm{b}z]\end{aligned}$$

Figures(B.2,B.3,B.4) show the schematics of the Lorenz systems and the coupling dynamics. The components are standard 5% tolerance resistors and 20% tolerance tantalum capacitors. No care was taken to match each component between the two systems, so there inevitably are differences between the two Lorenz systems at the component level. The Op-Amps are general purpose TL084's and the multipliers are Analog Devices AD633's. Since the output of the AD633 is internally divided by 10, the multiplication symbol shown in the schematic is actually an AD633 with a TL071 Op-Amp used for a gain of 10.

When taking data, the potentiometer in the data system is set manually to a fixed desired value, the potentiometer in the receiver system is set by the PC with the use of a stepper motor. The values of the parameter R may be calculated from

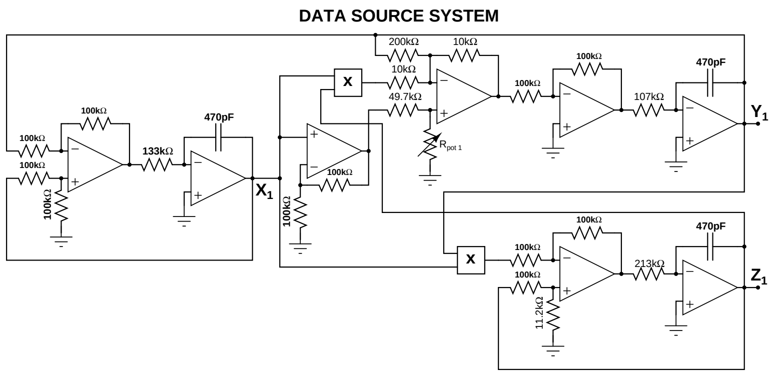


Figure B.2: Schematic of Lorenz “Data” circuit

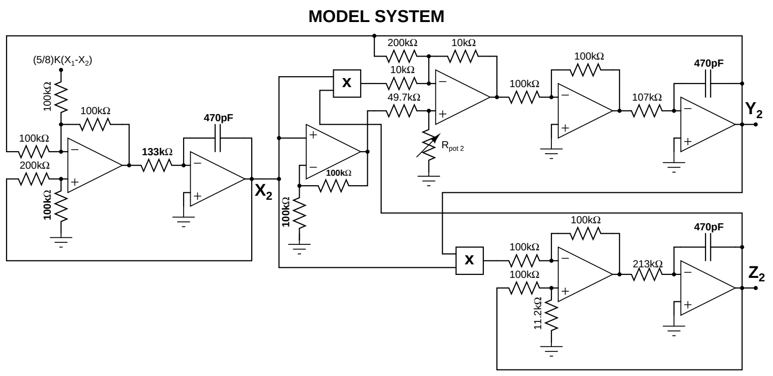


Figure B.3: Schematic of Lorenz “Receiver” circuit

the potentiometer values using the following equation.

$$R = \frac{82 \cdot R_{pot}}{R_{pot} + 49.7k\Omega}$$

The potentiometers are both 10-turn 100kΩ. Parameter scans use the range $20k\Omega \leq R_{pot} \leq 80k\Omega$ which translates to the parameter range $23.5 \leq R_2 \leq 50.6$ mentioned above. The equations describing the dynamics of the above electrical circuit are the following:

$$\begin{aligned}\dot{x_1} &= 1000[16(y_1 - x_1)] \\ \dot{y_1} &= 1000[R_1 x_1 - y_1 - 20 x_1 z_1]\end{aligned}$$

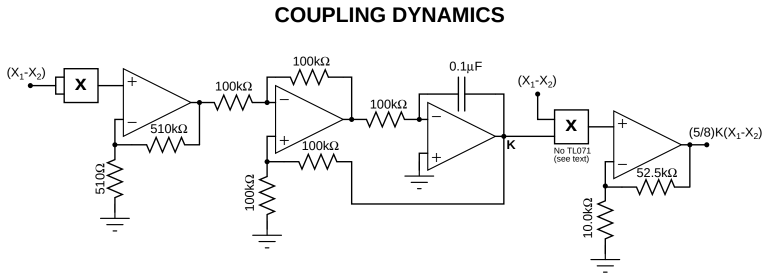


Figure B.4: Schematic of coupling K(t) circuit

$$\dot{z_1} = 1000[20x_1y_1 - 4z_1]$$

$$\dot{x_2} = 1000[16(y_2 - x_2) + 10K(x_1 - x_2)]$$
$$\dot{y_2} = 1000[\mathrm{R}_2x_2 - y_2 - 20x_2z_2]$$
$$\dot{z_2} = 1000[20x_2y_2 - 4z_2]$$

$$\dot{K} = 1000[-0.10K + 110(x_1 - x_2)^2]$$

Note that in this Appendix the receiver system has dynamical variables with subscript 2, so the coupling in the 'model' is $10K(x_1 - x_2)$.

Figure (B.5) shows a section of the time series of $x_1(t)$, $x_2(t)$ and $K(t)$ as measured by the PC. The Lorenz systems are creating the expected time series waveforms for $x_1(t)$ and $x_2(t)$ and these waveforms are for the most part synchronized. The $K(t)$ waveform is tending to decay toward zero with growth dependent on the spiking in the error signal $(x_1(t) - x_2(t))^2$. After acquiring 2 seconds of data sampled at 50kHz, N=100000 samples, the computer calculates the associated cost function based on the following formula:

$$\mathrm{C} = \frac{1}{\mathrm{N}} \sum_{j=1}^{\mathrm{N}} \tanh(\,200 \cdot (x_1(j) - x_2(j))^2\,)$$

The convergence of this cost function is verified in Figure(B.6) at four positions in the parameter scan ($\mathrm{R}_{\mathrm{pot},2}$ = 30,40,50,60 kΩ) with $\mathrm{R}_{\mathrm{pot},1} = 50k\Omega$. In each case

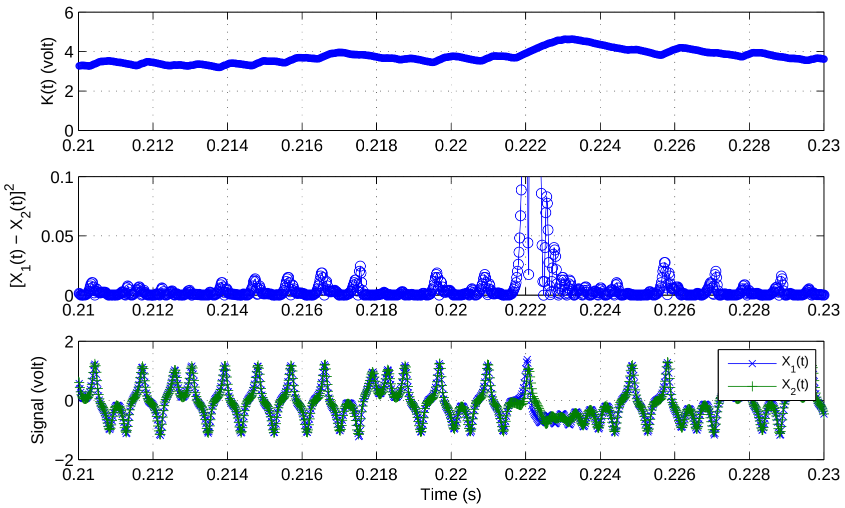


Figure B.5: Sample time series

the cost function converges to a steady value within the fist second of the 2 second sample window. The steady value of this cost function is automatically recorded over the range of $\mathrm{R}_2$ producing the parameter scan plots shown in Figure(B.7).

In the case shown in Figure(B.7), three different values of the data system parameter $\mathrm{R}_1$ were used. In each case there is clearly a minimum in the cost function in the neighborhood of $\mathrm{R}_2 = \mathrm{R}_1$. There are many sources of noise in this lab setup and it is not surprising that the minimum in the cost function is broader and shallower than the theoretical (noiseless) result - see Figure(B.8). Also, since the components used are of low tolerance, it is very likely that the system parameters that are assumed to be identical between the data system and the model ($\sigma$,b) are in fact not the same. Also, the value of R depends on several other components in addition to the potentiometer, so that equal settings on the potentiometers does not necessarily imply $\mathrm{R}_2 = \mathrm{R}_1$. Still, even with these imperfections, this experiment demonstrates that the dynamical coupling method ($K(t)$) does a good job of identifying a small neighborhood in which the presumably unknown parameter lies.

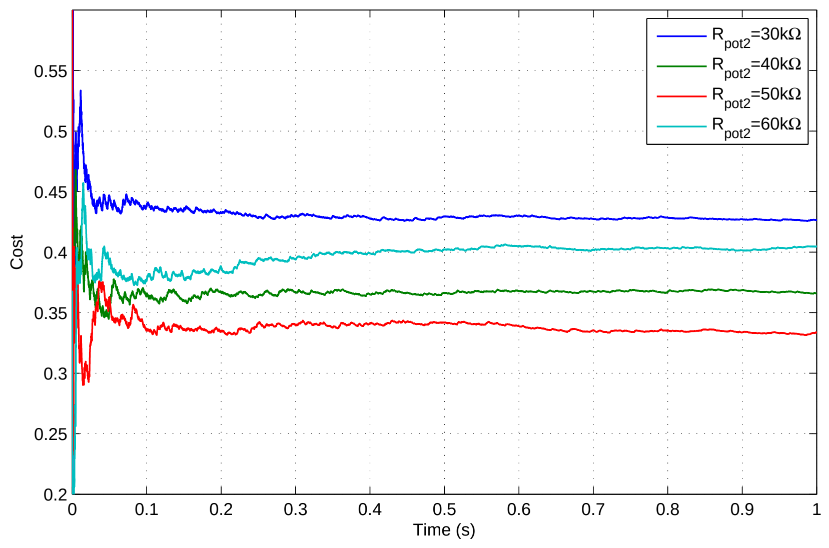


Figure B.6: Cost function convergence

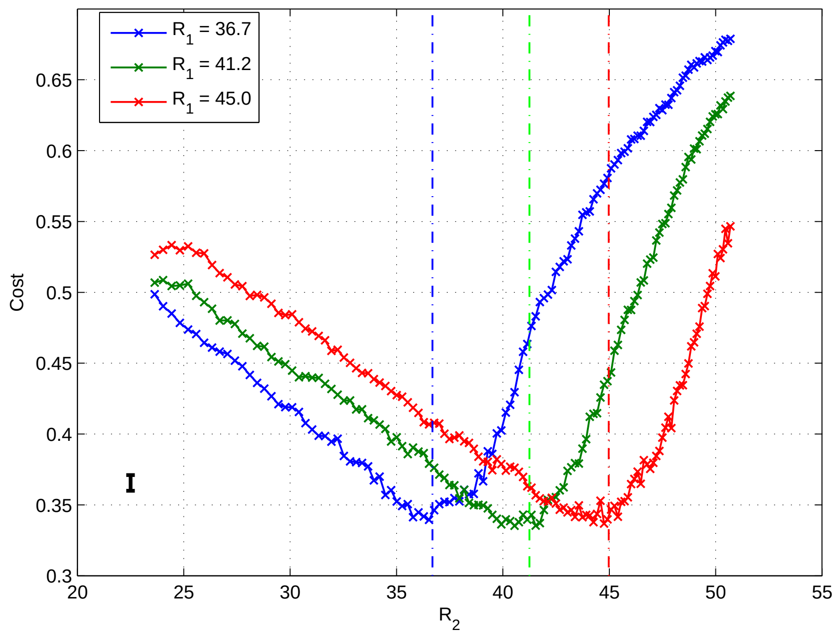


Figure B.7: Parameter scans

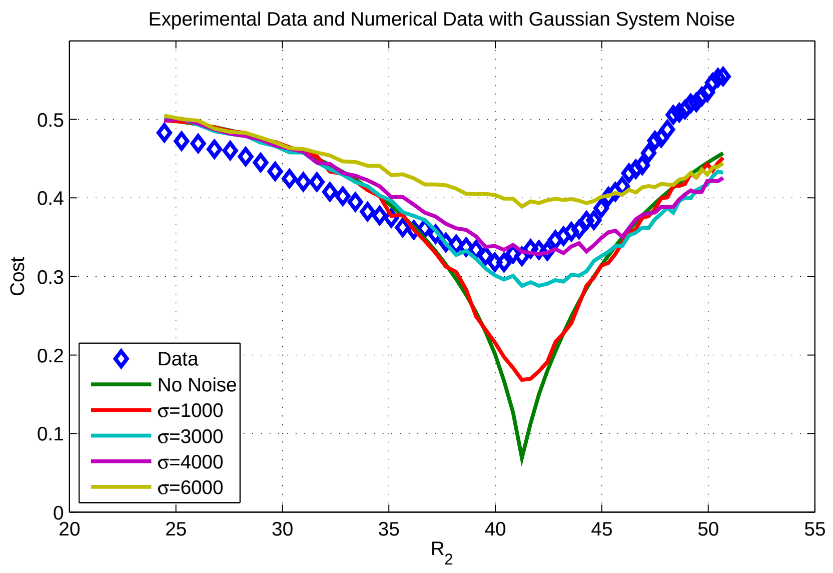


Figure B.8: Simulation with gaussian noise added to the state equations

# Appendix C

# Source Code

This section provides listings of relevant computer code.

## C.1 PIC Microcontroller Code

Assembly code used to program the parameter switching of the Lorenz circuits in chapter 4.

```
INCLUDE <p18f1320.inc>

__CONFIG  _CONFIG1H, 0xC8
__CONFIG  _CONFIG2H, 0x1E

;;;;;;;;;;;;;;;;;;;;;;;
;  VARIABLES
;
; State:  0x000 [1 byte]
;
; Counter:  0x001 [1 byte]
;
;;;;;;;;;;;;;;;;;;;;;;;

cblock 0x000
State, Counter
endc

#DEFINE pHEAD 0xA0
#DEFINE Ncount 0x07
; 300ms Timer
;#DEFINE TMRPREH 0xF6
;#DEFINE TMRPREL 0xD7
; 3 second Timer
#DEFINE TMRPREH 0xA4
#DEFINE TMRPREL 0x71

; Reset vector
org 0x00
bra Main

; High Priority Interrupt vector
org 0x08
bra ISR_Timer0

; Main code section
org 0x2A
Main

;;;;;;;;;;;;;;;;;;
; Setup PORT I/O ;
;;;;;;;;;;;;;;;;;;

; Configure Oscillator (8MHz Internal)
movlw 0x70
movwf OSCCON
```

```
clrf WDTCON

setf ADCON1

; PORTA [IIII OOOO]
movlw 0xF0
movwf TRISA
movlw 0x0F
movwf PORTA


; PORTB [OOIO OIII]
movlw 0x27
movwf TRISB
movlw 0xD0
movwf PORTB

;;;;;;;;;;;;;;;;;;
;  Setup Timer0  ;
;;;;;;;;;;;;;;;;;;

; Off, 16-bit mode, prescale 1:256
  movlw 0x07
movwf T0CON
bcf INTCON, TMR0IF

;;;;;;;;;;;;;;;;;;;;;;;;;
;  Configure Interrupts ;
;  & set initial state  ;
;;;;;;;;;;;;;;;;;;;;;;;;;

movlw 0xA0
movwf INTCON

clrf State
movlw Ncount
movwf Counter
bcf PORTB, RB4 ; Power LED ON
bsf PORTB, RB6 ; Running LED OFF
bcf PORTB, RB7 ; Ready LED ON

;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Load parameter pattern  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    lfsr FSR0, pHEAD
movlw b'00000010'
movwf POSTINC0
movlw b'00000100'
movwf POSTINC0
movlw b'00001000'
movwf POSTINC0
```

```
movlw b'00000100'
movwf POSTINC0
movlw b'00000010'
movwf POSTINC0
movlw b'00000001'
movwf POSTINC0
movlw b'00000001'
lfsr FSR0, pHEAD

Main_Loop

movlw 0x01
movwf PORTA
clrf State
movlw Ncount
movwf Counter
movlw TMRPREH
movwf TMR0H
movlw TMRPREL
movwf TMR0L
bcf INTCON, TMR0IF

WaitGO
btfss PORTB, RB1
bra WaitGO

bsf PORTB, RB7 ; Ready LED OFF
bcf PORTB, RB6 ; Running LED ON

; Send trigger pulse
bsf PORTB, RB3
nop
nop
bcf PORTB, RB3

bsf T0CON, TMR0ON

Running
btfss State, 0
bra Running

bsf PORTB, RB6 ; Running LED OFF
bcf PORTB, RB7 ; Ready LED ON

lfsr FSR0, pHEAD

bra Main_Loop

;;;;;;;;;;;;;;;;;;;;;;
;  High ISR Routine  ;
;;;;;;;;;;;;;;;;;;;;;;
;
```

```
; Executes every 300 ms
;
ISR_Timer0
movff POSTINC0, PORTA

bcf T0CON, TMR0ON
movlw TMRPREH
movwf TMR0H
movlw TMRPREL
movwf TMR0L
bsf T0CON, TMR0ON

decf Counter, F
btfss STATUS, Z
bra ISR_1

incf State, F
bcf T0CON, TMR0ON

ISR_1
bcf INTCON, TMR0IF
retfie 1

END
```

## C.2 Python Code

Python code used to implement the SNOPT optimal control method of parameter and state estimation of chapter 5. This code discretizes a given system using the Hermite-Simpson method with error feedback coupled to the first state equation. The optimization problem is defined and initialized for use by SNOPT and the optimization routine is called from the SNOPT FORTRAN library. In this example, the output is both saved to a file and plotted in a graphics window.

```
from numpy import *
from test7 import *
import sympy
import math
import pylab

print "-----------Hermite-Simpson ---------"
Feqnstr = []
UsrFcns = []
##################Hodgkin-Huxley###############
#
#   Vector Field
Feqnstr.append\
("gna*(x2**3)*x3*(vna-x1) - gk*(x4**4)*(vk+x1) - gl*(vl+x1) + 2.0")
Feqnstr.append\
("am*(1-x2) - bm*x2")
Feqnstr.append\
("ah*(1-x3) - bh*x3")
Feqnstr.append\
("an*(1-x4) - bn*x4")
#
#   User Functions
UsrFcns.append\
("am = -pm2*(pm1+x1) / ( sympy.exp(-pm2*(pm1+x1)) - 1.0 )")
UsrFcns.append\
("bm = sympy.exp(-(x1+pm3)/pm4)")
UsrFcns.append\
("ah = ph1*sympy.exp(-(ph2+x1)/ph3)")
UsrFcns.append\
("bh = 1.0 / ( sympy.exp(-ph5*(x1+ph4)) + 1.0 )")
UsrFcns.append\
("an = -pn1*(pn2+x1) / ( sympy.exp(-(pn2+x1)) - 1.0 )")
UsrFcns.append\
("bn = pn3*sympy.exp(-(x1+pn4)/pn5)")
#
#----------------STATE VARIABLES----------------------
#   names
Lvars = ["x1","x2","x3","x4"]
#
```

```
#   range [lower , upper]
#
Lvarlims = [[-200.0 , 200.0],\
            [0.0  , 1.0],\
            [0.0  , 1.0],\
            [0.0  , 1.0]]
#
#----------------PARAMETERS---------------------------
#   names
Lparams = ["gna","vna","gk","vk","gl","vl",\
           "pm1","pm2","pm3","pm4",\
           "ph1","ph2","ph3","ph4","ph5",\
           "pn1","pn2","pn3","pn4","pn5"]
#
#   range [lower , init , upper]
#
Lparamvals = [[10.0 , 20.0 , 40.0],\
              [25.0 , 50.0 , 100.0],\
              [3.00 , 6.00 , 12.0],\
              [50.0 , 100.0, 200.0],\
              [0.01 , 0.03 , 0.10],\
              [25.0 , 50.0 , 100.0],\
# Pm (1-4)
              [15.0 , 35.0 , 100.0],\
              [0.00 , 0.10 , 0.20],\
              [30.0 , 60.0 , 100.0],\
              [10.0 , 18.0 , 50.0],\
# Ph (1-5)
              [0.02 , 0.07 , 0.20],\
              [30.0 , 60.0 , 100.0],\
              [10.0 , 20.0 , 40.0],\
              [15.0 , 30.0 , 50.0],\
              [0.00 , 0.10 , 0.20],\
# Pn (1-5)
              [0.00 , 0.01 , 0.02],\
              [25.0 , 50.0 , 100.0],\
              [0.05 , 0.125, 0.25],\
              [30.0 , 60.0 , 100.0],\
              [50.0 , 80.0 , 150.0]]
#
#-----------------DATA FILE--------------------------
#
fName = "HH_data.dat"
hstep = 0.2
Nstop = 200
#
###############################################

Lcouple = ["k11","dk11"]
Fdim = len(Feqnstr)
Pdim = len(Lparams)
```

```
# Make symbols
Sv = []  # list of variable symbols
Sp = []  # list of parameter symbols
for i in range(len(Lvars)):
   Sv.append(sympy.Symbol(Lvars[i]))
for i in range(len(Lparams)):
   Sp.append(sympy.Symbol(Lparams[i]))
Sd = sympy.Symbol("myData")  # symbol for data(t)
Sk = [sympy.Symbol(Lcouple[0]), sympy.Symbol(Lcouple[1])]
Sall = Sv + Sk + Sp
print Sall
print "-------Symbols-----------"
print Sv,",",Sp,",",Sd,",",Sk

# Declare symbolic user functions
if len(UsrFcns) != 0:
   print "User Functions:"
   for k in range(len(UsrFcns)):
      sTemp1 = UsrFcns[k]
      for i in range(len(Lvars)):  # replace variables
         sTemp2 = "Sv[%d]" % i
         sTemp1 = sTemp1.replace(Lvars[i],sTemp2)
      for i in range(len(Lparams)):# replace parameters
         sTemp2 = "Sp[%d]" % i
         sTemp1 = sTemp1.replace(Lparams[i],sTemp2)
      exec sTemp1
      print sTemp1
else:
   print "No User Functions"
print "--------------------------------"

# Define symbolic vector field
Feqns = []
for k in range(Fdim):
   sTemp1 = Feqnstr[k]
   for i in range(len(Lvars)):  # replace variables
      sTemp2 = "Sv[%d]" % i
      sTemp1 = sTemp1.replace(Lvars[i],sTemp2)
   for i in range(len(Lparams)):# replace parameters
      sTemp2 = "Sp[%d]" % i
      sTemp1 = sTemp1.replace(Lparams[i],sTemp2)
   if k == 0:
      exec "F1=" + sTemp1
      sTemp1 += " + Sk[0]*(Sd-Sv[0])"
   sTemp2 = "Feqns.append("
   sTemp2 = sTemp2 + sTemp1 + ")"
   exec sTemp2

print "-------Vector Field-----------"
print "F1 = ",F1
for k in range(Fdim):
   print Feqns[k]
```

```
# Call the snInit routine
snopy.tset = Nstop
snopy.initpy()
T = int(snopy.t)

print "T = ",T
Xvec = zeros(((2*Fdim+3)*T - Fdim - 1 + Pdim),float)
print "x vector length = ",len(Xvec)

Tdata = []
X1data = []
X2data = []
X3data = []
X4data = []

fIn = open(fName, 'r')
sIn = fIn.readlines()
fIn.close()
for i in range(len(sIn)):
   Tdata.append(eval(sIn[i][1:12]))
   X1data.append(eval(sIn[i][13:25]))
   X2data.append(eval(sIn[i][26:38]))
   X3data.append(eval(sIn[i][39:51]))
   X4data.append(eval(sIn[i][52:64]))

print sIn[0]
print X1data[0],X2data[0],X3data[0],X4data[0]

Xinit = [X1data[0] , X2data[0] , X3data[0] , X4data[0]]

print "x vector length = ",len(Xvec)

Tlist = range(T-1)   # Runs from 0->T-2

AllCon = []
# Simpson Constraints
for k in range(len(Sv)):
   tCon = []
   tCon.append(Sv[k] + (hstep/6.0)*Feqns[k])
   tCon.append(-Sv[k] + (hstep/6.0)*Feqns[k])
   tCon.append((2.0*hstep/3.0)*Feqns[k])
   AllCon.append(tCon)
# Hermite Constraints
for k in range(len(Sv)):
   tCon = []
   tCon.append(0.5*Sv[k] + (hstep/8.0)*Feqns[k])
   tCon.append(0.5*Sv[k] - (hstep/8.0)*Feqns[k])
   tCon.append(-Sv[k])
   AllCon.append(tCon)
# Hermite Control Constraint
tCon = []
```

```
tCon.append(0.5*Sk[0] + (hstep/8.0)*Sk[1])
tCon.append(0.5*Sk[0] - (hstep/8.0)*Sk[1])
tCon.append(-Sk[0])
AllCon.append(tCon)

# find linear and nonlinear variables
def linorno(mySym):
   dF = sympy.diff(F,mySym)
   if dF == 0:
      return 0
   else:
      myTemp = 1
      for j in range(len(Sv)):  # variables
         d2F = sympy.diff(dF,Sv[j])
         if d2F != 0:
            myTemp = 2
      for j in range(len(Sp)):  # parameters
         d2F = sympy.diff(dF,Sp[j])
         if d2F != 0:
            myTemp = 2
      d2F = sympy.diff(dF,Sk[0])  # control
      if d2F != 0:
         myTemp = 2
   return myTemp

Mv = []
for k in range(len(AllCon)):  # over all constraints
   tList = []
   F = AllCon[k][0]
   for i in range(len(Sall)):  # over all symbols
      tList.append(linorno(Sall[i]))
   Mv.append(tList)

print "----- 0:N/A --- 1:Linear --- 2:Nonlinear -----"
for k in range(len(Mv)):
   print Mv[k]

dict1 = {0:"Xval",1:"Xvalp1",2:"Xval2"}
Xval = zeros(len(Lvars),float)
Xvalp1 = zeros(len(Lvars),float)
Xval2 = zeros(len(Lvars),float)
Pval = zeros(len(Lparams),float)
dict2 = {0:"K11val",1:"K11valp1",2:"K11val2"}
K11val = 0.0
K11valp1 = 0.0
K11val2 = 0.0
dict3 = {0:"dK11val",1:"dK11valp1",2:""}
dK11val = 0.0
dK11valp1 = 0.0
dict4 = {0:"Xdval",1:"Xdvalp1",2:"Xdval2"}
Xdval = 0.0
Xdvalp1 = 0.0
```

```
Xdval2 = 0.0
ftemp = 0.0
CFobj = 0.0

def subvars(mystr,myi):
   mytemp = mystr
   for j in range(len(Sv)):
      Srep = dict1[myi] + "[%d]" % j
      Sfind = Lvars[j]
      mytemp = mytemp.replace(Sfind,Srep)

      Srep = dict2[myi]
      Sfind = Lcouple[0]
      mytemp = mytemp.replace(Sfind,Srep)

      Srep = dict3[myi]
      Sfind = Lcouple[1]
      mytemp = mytemp.replace(Sfind,Srep)

   for j in range(len(Sp)):
      Srep = "Pval[%d]" % j
      Sfind = Lparams[j]
      mytemp = mytemp.replace(Sfind,Srep)
   Srep = dict4[myi]
   Sfind = "myData"
   mytemp = mytemp.replace(Sfind,Srep)
   return mytemp

# Build constraint equation strings
strAllCon = []
for icon in range(len(AllCon)):
   temp1 = []
   for n in [0,1,2]:
      Stemp = repr(AllCon[icon][n])
      Stemp = subvars(Stemp,n)
      temp1.append(Stemp)
   strAllCon.append(temp1)


Stemp = repr(F1)
Stemp = subvars(Stemp,0)
strF1=Stemp
#print strF1

# Build Jacobian strings
sJac = []
for icon in range(len(AllCon)):
   temp1 = []
   for jvar in range(len(Sall)):
      temp2 = []
      for n in [0,1,2]:
         Stemp = repr(sympy.diff(AllCon[icon][n],Sall[jvar]))
         Stemp = subvars(Stemp,n)
         temp2.append(Stemp)
      temp1.append(temp2)
   sJac.append(temp1)

Tlist = range(T-1)   # Runs from 0->T-2
Avec = []
iAf = []
jAv = []
Gvec = []
iGf = []
jGv = []
K11idx = Fdim*T
dK11idx = (Fdim+1)*T
Xn2idx = (Fdim+2)*T
K112idx = (2*Fdim + 2)*T - Fdim
Paramidx = (2*Fdim + 3)*T - Fdim - 1

for nt in Tlist:   # Over Time 0 -> T-2
   for k in range(len(AllCon)):  # Over Constraints
      for i in range(len(Sall)):  # Over Symbols
# State Variables
         if i < Fdim:
            if Mv[k][i] == 1:
               # n
               iAf.append(k*T + nt + 1)
               jAv.append(i*T + nt + 1)
               Avec.append(sympy.diff(AllCon[k][0],Sall[i]).evalf())
               # n+1
               iAf.append(k*T + nt + 1)
               jAv.append(i*T + nt + 2)
               Avec.append(sympy.diff(AllCon[k][1],Sall[i]).evalf())
               # n2
               iAf.append(k*T + nt + 1)
               jAv.append(i*(T-1) + nt + 1 + Xn2idx)
               Avec.append(sympy.diff(AllCon[k][2],Sall[i]).evalf())
            elif Mv[k][i] == 2:
               # n
               iGf.append(k*T + nt + 1)
               jGv.append(i*T + nt + 1)
               Gvec.append([0,k,i,nt])
               # n+1
               iGf.append(k*T + nt + 1)
               jGv.append(i*T + nt + 2)
               Gvec.append([1,k,i,nt])
               # n2
               if k in range(len(Sv)):
                  iGf.append(k*T + nt + 1)
                  jGv.append(i*(T-1) + nt + 1 + Xn2idx)
                  Gvec.append([2,k,i,nt])
               else:
                  iAf.append(k*T + nt + 1)
                  jAv.append(i*(T-1) + nt + 1 + Xn2idx)
                  Avec.append\
                  (sympy.diff(AllCon[k][2],Sall[i]).evalf())
# K11,K11p1,K112
         elif i == Fdim:
            if Mv[k][i] == 1:
               # n
               iAf.append(k*T + nt + 1)
               jAv.append(nt + K11idx + 1)
               Avec.append(sympy.diff(AllCon[k][0],Sall[i]).evalf())
               # n+1
               iAf.append(k*T + nt + 1)
               jAv.append(nt + K11idx + 2)
               Avec.append(sympy.diff(AllCon[k][1],Sall[i]).evalf())
               # n2
               iAf.append(k*T + nt + 1)
               jAv.append(nt + K112idx + 1)
               Avec.append(sympy.diff(AllCon[k][2],Sall[i]).evalf())
            elif Mv[k][i] == 2:
               # n
               iGf.append(k*T + nt + 1)
               jGv.append(nt + K11idx + 1)
               Gvec.append([0,k,i,nt])
               # n+1
               iGf.append(k*T + nt + 1)
               jGv.append(nt + K11idx + 2)
               Gvec.append([1,k,i,nt])
               # n2
               if k in range(len(Sv)):
                  iGf.append(k*T + nt + 1)
                  jGv.append(nt + K112idx + 1)
                  Gvec.append([2,k,i,nt])
# dK11
         elif i == Fdim+1:
            if Mv[k][i] == 1:
               # n
               iAf.append(k*T + nt + 1)
               jAv.append(nt + dK11idx + 1)
               Avec.append(sympy.diff(AllCon[k][0],Sall[i]).evalf())
               # n+1
               iAf.append(k*T + nt + 1)
               jAv.append(nt + dK11idx + 2)
               Avec.append(sympy.diff(AllCon[k][1],Sall[i]).evalf())
            elif Mv[k][i] == 2:
               # n
               iGf.append(k*T + nt + 1)
               jGv.append(nt + dK11idx + 1)
               Gvec.append([0,k,i,nt])
               # n+1
               iGf.append(k*T + nt + 1)
               jGv.append(nt + dK11idx + 2)
               Gvec.append([1,k,i,nt])
```

```
# Parameters
         else:
            if Mv[k][i] == 1:
               iAf.append(k*T + nt + 1)
               jAv.append(Paramidx + (i-Fdim-2) + 1)
               Avec.append(hstep)  # dCk/dP
            elif Mv[k][i] == 2:
               iGf.append(k*T + nt + 1)
               jGv.append(Paramidx + (i-Fdim-2) + 1)
               Gvec.append([-1,k,i,nt])


print "Length of iAf = ",len(iAf)

print "Length of G w/o objective = ",len(iGf)
lenGsave = len(iGf)
print "Length of AllCon",len(AllCon)
# Add on Objective components of G
for i in range(T):
   iGf.append(len(AllCon)*T + 1)
   jGv.append(i + 1)
   iGf.append(len(AllCon)*T + 1)
   jGv.append(i + K11idx + 1)
   iGf.append(len(AllCon)*T + 1)
   jGv.append(i + dK11idx + 1)
lenGsave2 = len(iGf)
for i in range(T-1):
   iGf.append(len(AllCon)*T + 1)
   jGv.append(i + Xn2idx + 1)
   iGf.append(len(AllCon)*T + 1)
   jGv.append(i + K112idx + 1)

print "Length of iGf = ",len(iGf)

# Load the spDat data into snopy module
snopy.n = len(Xvec)
snopy.nf = T*len(AllCon) + 1
snopy.objrow = T*len(AllCon) + 1
snopy.objadd = 0.0
snopy.nea = len(iAf)
snopy.neg = len(iGf)
for i in range(len(iAf)):
   snopy.iafun[i] = iAf[i]
   snopy.javar[i] = jAv[i]
   snopy.a[i] = Avec[i]
for i in range(len(iGf)):
   snopy.igfun[i] = iGf[i]
   snopy.jgvar[i] = jGv[i]
# Initialize Xn,Xn2
for i in range(T):
   for j in range(len(Sv)):
      snopy.x[j*T + i] = 0.0
      snopy.xlow[j*T + i] = Lvarlims[j][0]
      snopy.xupp[j*T + i] = Lvarlims[j][1]
      snopy.xstate[j*T + i] = 0
      if i < (T-1):
         snopy.x[j*(T-1) + i + Xn2idx] = 0.0
         snopy.xlow[j*(T-1) + i + Xn2idx] = Lvarlims[j][0]
         snopy.xupp[j*(T-1) + i + Xn2idx] = Lvarlims[j][1]
         snopy.xstate[j*(T-1) + i + Xn2idx] = 0
# Initialize Kn,dKn,Kn2
for i in range(T):
   for j in [0,1]:
      snopy.x[j*T + i + K11idx] = 0.01
      snopy.xlow[j*T + i + K11idx] = 0.0
      snopy.xupp[j*T + i + K11idx] = 100.0
      snopy.xstate[j*T + i + K11idx] = 0
   if i < (T-1):
      snopy.x[i + K112idx] = 0.01
      snopy.xlow[i + K112idx] = 0.0
      snopy.xupp[i + K112idx] = 100.0
      snopy.xstate[i + K112idx] = 0
# Initialize Params
for i in range(len(Lparamvals)):
   snopy.xlow[Paramidx+i] = Lparamvals[i][0]
   snopy.x[Paramidx+i] = Lparamvals[i][1]
   snopy.xupp[Paramidx+i] = Lparamvals[i][2]
   snopy.xstate[Paramidx+i] = 3

for i in range(T*len(AllCon)):
   snopy.fmul[i]=0.0
   snopy.flow[i]=0.0
   snopy.fupp[i]=0.0
snopy.fupp[T*len(AllCon)] = 1e20

for i in range(T):
   snopy.x[i] = X1data[2*i]
   snopy.x[i + Xn2idx] = X1data[2*i+1]

# Define callback function pyfillfg
def pyfg2(a1,a2,myx,needf,a5,myf,needg,a8,
          myg,a10,a11,a12,a13,a14,a15):
   for j in range(len(Sp)):
      Pval[j] = myx[Paramidx+j]
   if needg > 0:
      for i in range(lenGsave):
         nidx = Gvec[i][0]
         kidx = Gvec[i][1]
         iidx = Gvec[i][2]
         tidx = Gvec[i][3]
         for j in range(len(Sv)):
            Xval[j] = myx[(j*T) + tidx]
            Xvalp1[j] = myx[(j*T) + tidx + 1]
            Xval2[j] = myx[(j*(T-1)) + Xn2idx + tidx]
         K11val = myx[K11idx + tidx]
         K11valp1 = myx[K11idx + tidx + 1]
         K11val2 = myx[K112idx + tidx]
         dK11val = myx[dK11idx + tidx]
         dK11valp1 = myx[dK11idx + tidx + 1]
         Xdval = X1data[2*tidx]
         Xdvalp1 = X1data[2*tidx+2]
         Xdval2 = X1data[2*tidx+1]
         ftemp = 0.0
         if nidx == -1:
            ftemp += eval(sJac[kidx][iidx][0])
            ftemp += eval(sJac[kidx][iidx][1])
            ftemp += eval(sJac[kidx][iidx][2])
         else:
            ftemp += eval(sJac[kidx][iidx][nidx])
         myg[i] = ftemp
      for i in range(T):
         myg[lenGsave + 3*i] = myx[i] - X1data[2*i]
         myg[lenGsave + 3*i+1] = myx[i + K11idx]
         myg[lenGsave + 3*i+2] = myx[i + dK11idx]
      for i in range(T-1):
         myg[lenGsave2 + 2*i] = myx[i+Xn2idx] - X1data[2*i+1]
         myg[lenGsave2 + 2*i+1] = myx[i + K112idx]

   if needf > 0:
      CFobj = 0.0
      for tidx in Tlist:   # Over Time
         K11val = myx[K11idx + tidx]
         K11valp1 = myx[K11idx + tidx + 1]
         K11val2 = myx[K112idx + tidx]
         dK11val = myx[dK11idx + tidx]
         Xdval = X1data[2*tidx]
         Xdvalp1 = X1data[2*tidx+2]
         Xdval2 = X1data[2*tidx+1]
         for k in range(2*len(Sv)):  # Over Constraints
            for i in range(len(Sv)):  # Over Variables
               if Mv[k][i] == 2:
                  Xval[i] = myx[(i*T) + tidx]
                  Xvalp1[i] = myx[(i*T) + tidx + 1]
                  Xval2[i] = myx[(i*(T-1)) + Xn2idx + tidx]
                  if k in [4,5,6,7]: Xval2[i] = 0.0
               else:
                  Xval[i] = 0.0
                  Xvalp1[i] = 0.0
                  Xval2[i] = 0.0
            ftemp = 0.0
            for i in [0,1,2]:
               ftemp += eval(strAllCon[k][i])
            myf[k*T+tidx] = ftemp
         CFobj += (myx[tidx] - Xdval)**2 + K11val**2 + dK11val**2
         if tidx < T-1:
            CFobj += (myx[Xn2idx+tidx] - Xdval2)**2 + K11val2**2
      myf[len(AllCon)*T] = CFobj / 2.0

   if needf == 0 and needg == 0:
      print "PYTHON: This was the first call"

   return

# Call snOptA
snopy.snoptapy(pyfg2)

# Get the final result from x
for i in range(len(Xvec)):
   Xvec[i] = snopy.x[i]

# Close files
snopy.closepy()

# Print out parameter values
for i in range(len(Lparams)):
   Pval[i] = Xvec[Paramidx+i]
   print Lparams[i] + " = " + str(Pval[i])

# Plot the output : Requires matplotlib and tk
x1val = []
x1dat = []

x2val = []
x2dat = []

x3val = []
x3dat = []

x4val = []
x4dat = []

kval = []
tval = []

R1 = []
R2 = []
ftemp1 = 0.0
ftemp2 = 0.0

for i in Tlist:
   x1val.append(Xvec[i])
   Xval[0]=Xvec[i]
   x2val.append(Xvec[T+i])
   Xval[1]=Xvec[T+i]
   x3val.append(Xvec[2*T+i])
   Xval[2]=Xvec[2*T+i]
   x4val.append(Xvec[3*T+i])
   Xval[3]=Xvec[3*T+i]
   kval.append(Xvec[K11idx+i])
   Kval=Xvec[K11idx+i]
   ftemp1 = eval(strF1)
   ftemp2 = Kval*(X1data[2*i] - Xval[0])
   R1.append( ftemp1**2 / (ftemp1**2 + ftemp2**2) )
   R2.append( ftemp1**2 / (ftemp1**2 + Kval**2) )

   x1val.append(Xvec[i + Xn2idx])
   Xval[0]=Xvec[i + Xn2idx]
   x2val.append(Xvec[i + Xn2idx + (T-1)])
   Xval[1]=Xvec[i + Xn2idx + (T-1)]
   x3val.append(Xvec[i + Xn2idx + 2*(T-1)])
   Xval[2]=Xvec[i + Xn2idx + 2*(T-1)]
   x4val.append(Xvec[i + Xn2idx + 3*(T-1)])
   Xval[3]=Xvec[i + Xn2idx + 3*(T-1)]
   kval.append(Xvec[K112idx+i])
   Kval=Xvec[K112idx+i]
   ftemp1 = eval(strF1)
   ftemp2 = Kval*(X1data[2*i+1] - Xval[0])
   R1.append( ftemp1**2 / (ftemp1**2 + ftemp2**2) )
   R2.append( ftemp1**2 / (ftemp1**2 + Kval**2) )

x1dat = X1data[:2*T-2]
x2dat = X2data[:2*T-2]
x3dat = X3data[:2*T-2]
x4dat = X4data[:2*T-2]
tval = Tdata[:2*T-2]

print "X1val length = ",len(x1val)
print "X1dat length = ",len(x1dat)
print "tval length = ",len(tval)

print "Writing File : neuron_out.dat"
myfile = open('neuron_out.dat','w')
for i in range(len(tval)):
   myStr = "%f,%f,%f,%f,%f,%f,%f,%f,%f,%f\n" %
           (tval[i],kval[i],x1val[i],x2val[i],x3val[i],
            x4val[i],x1dat[i],x2dat[i],x3dat[i],x4dat[i])
   myfile.write(myStr)
myfile.close()

exit()
```

## C.3 SPICE Code

The circuit simulation software SPICE (cite) was used to initialize and verify the parameters of the gating variable dynamics in the Hodgkin-Huxley electrical circuit. The code describing the circuits for $\alpha_{\mathrm{m,n,h}}$ and $\beta_{\mathrm{m,n,h}}$ are shown below.

### C.3.1 Code for $\alpha_{\mathrm{m}}(\mathrm{V}_m)$, $\beta_{\mathrm{m}}(\mathrm{V}_m)$

```
Hodgkin-Huxley Circuit m(V)

.include UA741.301

* Power Supplies
Vcc 100 0 DC 15
Vee 101 0 DC -15

* Membrane Voltage
Vin 1 0 DC 1.0

*****************
* Alpha_m Stage *
*****************

*Sources
Vath 2 0 DC -0.15
Va1 7 0 DC 0.3
Va2 8 0 DC -1.0
Vatr 12 0 DC 0.7

* Resistors
R1 1 3 470K
R2 2 3 470K
R5 3 4 560K
R31 4 5 1K
R32 4 6 1K
R33 5 7 1K
R34 6 8 1K
R35 4 11 470K
R36 9 11 47.5K
R37 10 11 22K
R40 11 12 470K
R42 11 13 43.2K

*Diodes
D1 5 9 D1N914
D2 6 10 D1N914

* Op-Amps
X1 0  3 100 101  4 UA741
```

```
X2 0 11 100 101 13 UA741

****************
* Beta_m Stage *
****************
*Sources
Vbth 14 0 DC -0.5
Vb1 19 0 DC -1.5
Vb2 20 0 DC -2.5
Vbtr 24 0 DC -4.5

* Resistors
R6  1 15 470K
R7 14 15 470K
R10 15 16 470K
R44 16 17 1K
R45 16 18 1K
R46 17 19 1K
R47 18 20 1K
R48 16 23 470K
R49 21 23 10K
R50 22 23 22K
R53 24 23 470K
R55 23 25 12K

*Diodes
D3 21 17 D1N914
D4 22 18 D1N914

* Op-Amps
X3 0 15 100 101 16 UA741
X4 0 23 100 101 25 UA741

.MODEL D1N914 D(IS=100E-15 RS=16 CJO=2PF
                TT=12NS BV=100 IBV=100E-15)

.dc Vin -3.75 1.25 0.01
.print dc V(13) V(25)
.save dc V(13) V(25)
.end
```

### C.3.2 Code for $\alpha_{\mathrm{n}}(\mathrm{V}_m)$, $\beta_{\mathrm{n}}(\mathrm{V}_m)$

```
Hodgkin-Huxley Circuit n(V)

.include UA741.301

* Power Supplies
Vcc 100 0 DC 15
Vee 101 0 DC -15
```

```
* Membrane Voltage
Vin 1 0 DC 1.0

*****************
* Alpha_n Stage *
*****************

*Sources
Vath 2 0 DC -0.611
Va1 7 0 DC 0.0
Va2 8 0 DC -1.0
Vatr 12 0 DC -1.54

* Resistors
R21 1 3 470K
R22 2 3 470K
R25 3 4 470K
R85 4 5 1K
R86 4 6 1K
R87 5 7 1K
R88 6 8 1K
R89 4 11 178K
R90 9 11 33K
R91 10 11 27K
R94 11 12 470K
R96 11 13 5.1K

*Diodes
D1 5 9 D1N914
D2 6 10 D1N914

* Op-Amps
X1 0  3 100 101  4 UA741
X2 0 11 100 101 13 UA741

****************
* Beta_n Stage *
****************
*Sources
Vbth 14 0 DC -0.48
Vb1 19 0 DC -1.5
Vb2 20 0 DC -2.5
Vbtr 24 0 DC -9.15

* Resistors
R26  1 15 470K
R27 14 15 470K
R30 15 16 470K
R98 16 17 1K
R99 16 18 1K
R100 17 19 1K
R101 18 20 1K
```

```
R102 16 23 470K
R103 21 23 178K
R104 22 23 178K
R107 24 23 470K
R109 23 25 56K

*Diodes
D3 21 17 D1N914
D4 22 18 D1N914

* Op-Amps
X3 0 15 100 101 16 UA741
X4 0 23 100 101 25 UA741

.MODEL D1N914 D(IS=100E-15 RS=16 CJO=2PF
                TT=12NS BV=100 IBV=100E-15)

.dc Vin -3.75 1.25 0.01
.print dc V(13) V(25)
.save dc V(13) V(25)
.end
```

### C.3.3 Code for $\alpha_{\mathrm{h}}(\mathrm{V}_m)$, $\beta_{\mathrm{h}}(\mathrm{V}_m)$

```
Hodgkin-Huxley Circuit h(V)

.include UA741.301

* Power Supplies
Vcc 100 0 DC 15
Vee 101 0 DC -15

* Membrane Voltage
Vin 1 0 DC 1.0

*****************
* Alpha_h Stage *
*****************

*Sources
Vath 2 0 DC -0.686
Va1 7 0 DC -1.5
Va2 8 0 DC -2.5
Vatr 12 0 DC -5.0

* Resistors
R11 1 3 470K
R12 2 3 470K
R15 3 4 470K
R57 4 5 1K
R58 4 6 1K
```

```
R59 5 7 1K
R60 6 8 1K
R61 4 11 470K
R62 9 11 10K
R63 10 11 27K
R66 11 12 470K
R68 11 13 22K

*Diodes
D1 9 5 D1N914
D2 10 6 D1N914

* Op-Amps
X1 0  3 100 101  4 UA741
X2 0 11 100 101 13 UA741

****************
* Beta_h Stage *
****************
*Sources
Vbth 14 0 DC -0.557
Vb1 19 0 DC 0.3
Vb2 20 0 DC -1.0
Vbtr 24 0 DC -0.362

* Resistors
R16  1 15 470K
R17 14 15 470K
R20 15 16 510K
R70 16 17 1K
R71 16 18 1K
R72 17 19 1K
R73 18 20 1K
R74 16 23 825K
R75 21 23 82K
R76 22 23 27K
R79 24 23 470K
R81 23 25 20K
R82 26 25 1K
R84 26 0 0.75K

*Diodes
D3 17 21 D1N914
D4 18 22 D1N914
D5 23 26 D1N914

* Op-Amps
X3 0 15 100 101 16 UA741
X4 0 23 100 101 25 UA741

.MODEL D1N914 D(IS=100E-15 RS=16 CJO=2PF
                TT=12NS BV=100 IBV=100E-15)
```

```
.dc Vin -3.75 1.25 0.01
.print dc V(13) V(25)
.save dc V(13) V(25)
.end
```

# Bibliography

[1] H. D. I. Abarbanel. *Analysis of Observed Chaotic Data*. Springer-Verlag, NY, 1996.

[2] H. D. I. Abarbanel, N. F. Rulkov, and M. M. Sushchik. Generalized synchronization of chaos: The auxiliary system approach. *Physical Review E*, 53:4528–4535, 1996.

[3] V. S. Afraimovich, N. N. Verichev, and M. I. Rabinovich. Stochastic synchronization of oscillators in dissipative systems. *Inv. VUZ Radiofiz.*, 29, 1986.

[4] G. Ananthakrishna. Current theoretical approaches to collective behavior of dislocations. *Physics Reports*, 440:113–259, 2007.

[5] G. Ananthakrishna and M. S. Bharathi. Dynamical approach to the spatiotemporal aspects of the portevin - le chatelier effect: Chaos, turbulence, and band propagation. *Physical Review E*, 70:026111, 2004.

[6] A. E. Bryson and Y-C Ho. *Applied Optimal Control: Optimization, Estimation, and Control.* Hemisphere, 1975.

[7] K. M. Cuomo, A. V. Oppenheim, and S. H. Strogatz. Synchronization of lorenz-based chaotic circuits with applications to communications. *IEEE Transactions, Circuits and Systems II*, 40:626–633, 1993.

[8] H. Dedieu and M. J. Ogorzalek. Identifiability and identification of chaotic systems based on adaptive synchronization. *IEEE Transactions, Circuits and Systems I*, 44:948–962, 1997.

[9] S. V. Franklin, F. Mertens, and M. Marder. Portevin–le chatelier effect. *Physical Review E*, 62, 2000.

[10] R. G. Gallager. *Information Theory and Reliable Communication*. Wiley, New York, 1968.

[11] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47:99–131, 2005.

[12] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[13] D. Huang. Synchronization-based estimation of all parameters of chaotic systems from time series. *Physical Review E*, 69:067201, 2004.

[14] D. Johnston and S. Wu. *Foundations of Cellular Neurophysiology.* MIT Press, 1995.

[15] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis.* Cambridge University Press, second edition, 2003.

[16] M. P. Kennedy. Chaos in the colpitts oscillator. *IEEE Transactions, Circuits and Systems I*, 41:771–774, 1994.

[17] D. E. Kirk. *Optimal Control Theory: An Introduction.* Prentice-Hall, NJ, 1970.

[18] R. Konnur. Synchronization-based approach for estimating all model parameters of chaotic systems. *Physical Review E*, 67:027204, 2003.

[19] H. P. Langtangen. *Python Scripting for Computational Science.* Springer, second edition, 2004.

[20] F. L. Lewis and V. L. Syrmos. *Optimal Control.* Wiley, second edition, 1995.

[21] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[22] M. Lutz and D. Ascher. *Learning Python.* O'Reilly, second edition, 2003.

[23] G. M. Maggio, O. D. Feo, and M. P. Kennedy. Nonlinear analysis of the colpitts oscillator and applications to design. *IEEE Transactions, Circuits and Systems I*, 46:1118–1130, 1999.

[24] A. Maybhate and R. E. Amriktar. Use of synchronization and adaptive control in parameter estimation from a time series. *Physical Review E*, 59:284–293, 1999.

[25] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Physical Review Letters*, 45, 1980.

[26] U. Parlitz, L. Junge, and L. Kocarev. Synchronization-based parameter estimation from time series. *Physical Review E*, 54:6253–6259, 1996.

[27] L. M. Pecora and T. L. Carroll. Synchronization in chaotic systems. *Physical Review Letters*, 64:821–824, 1990.

[28] A. Pikovsky, M. Rosenblum, and J. Kurths. *Synchronization : A universal concept in nonlinear sciences.* Cambridge University Press, 2001.

[29] N. F. Rulkov, M. M. Sushchik, L. S. Tsimring, and H. D. I. Abarbanel. Generalized synchronization of chaos in directionally coupled chaotic systems. *Physical Review E*, 51:980–994, 1995.

[30] H. Sakaguchi. Parameter evaluation from time sequences using chaos synchronization. *Physical Review E*, 65:027201, 2002.

[31] B. Saltzman. Finite amplitude free convection as an initial value problem. *Journal of the Atmospheric Sciences*, 19:329–341, 1962.

[32] T. Sauer, J. A. Yorke, and M. Casdagli. Embedology. *Journal of Statistical Physics*, 65, 1991.

[33] A. S. Sedra and K. C. Smith. *Microelectronic Circuits.* Oxford University Press, third edition, 1991.

[34] J. C. Sprott. *Chaos and Time–Series Analysis.* Oxford, 2003.

[35] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, 1986.

[36] D. J. Struik. *Lectures on Classical Differential Geometry.* Dover, second edition, 1988.

[37] F. Takens. Detecting strange attractors in turbulence. In D. Rand and L. S. Young, editors, *Dynamical Systems and Turbulence*, volume 898, page 366. Springer, 1981.

[38] D. Y. Tang and N. R. Heckenberg. Synchronization of mutually coupled chaotic systems. *Physical Review E*, 55:6618–6623, 1997.

[39] I. Tokuda, U. Parlitz, L. Illing, M. B. Kennel, and H. D. I. Abarbanel. Parameter estimation for neuron models. In *Proceedings of the 7th Experimental Chaos Conference*, San Diego, CA, USA, 2002.

[40] H. U. Voss, J. Timmer, and J. Kurths. Nonlinear dynamical system identification from uncertain and indirect measurements. *Int. J. Bif. Chaos*, 14:1905–1933, 2004.

[41] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determining lyapunov exponents from a time series. *Physica D*, 16:285–317, 1985.

[42] T. Yamada and H. Fujisaka. Stability theory of synchronized motion in coupled oscillator systems. *Pregress Theor. Phys.*, 70, 1983.